keyon

Unleash Microsoft PKI

keyon / true-Xtender Policy Module V2.3.3 User Guide

V1.1

December 2014

Copyright © 2014 by keyon AG

All rights reserved. No part of the contents of this manual may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Trademark Notice

keyon is a registered trademark of keyon AG in Switzerland and/or other countries.

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Table of contents

Overview 8
What it is the keyon / true-Xtender? 8
Key Features 8
Request processing with true-Xtender 9
Installation 10
Configure Certificate Services for keyon / true-Xtender 11
Backup and restore 14
Creating a backup of the current true-Xtender configuration 14
Restoring a true-Xtender configuration from a backup 15
Configuration 16
Using true-Xtender configuration application 16
Configure within Certificate Services Management 17
Template Processing 19
Enterprise CA Types 19
Standalone CA Types 20
Managing standalone templates 20
The default template 20
Adding custom templates 21
Removing custom templates 21
Specifying the template to use for a request with a Standalone CA 21
New Requests 21
Pending Requests 22
Delegate Policy Configuration 22
License 24
About 25
Template Configuration 26
Certificate Template Info 26
Enabling or disabling processing of this template 26
Subject DN Building 27
Directory Information Tree, RDNs and Distinguished Names 28
Adding a RDN element to the subject DN 30

Using static elements for RDN elements 30
Using the content of request attributes for RDN elements 31
Using the content of request RDNs 32
Removing a RDN element 34
Edit a RDN element 34
Reorder RDN elements 34
Samples 35
Keep request DN 35
Using original request RDN element 35
Use request attributes 36
Externally build DN 37
Custom X.509 Extensions 38
Adding custom extensions 38
Edit custom extension files 39
Removing custom extensions 40
Custom extensions definition language 40
Illustrated custom extension example 41
Root Tag 45
extension 45
Variable Related Tags 47
define 47
undefine 49
insert 50
inserttoken 51
Logic Tags 52
if 52
ifnot 54
ifcontains 56
ifnotcontains 58
exception 60
Loop Tags 61
foreach 61
foreachrequestrdn 63

foreachsubjectrdn 65
foreachsubjectaltname 67
Request Data Tags 69
requestattribute 69
definerequestattribute 70
requestproperty 71
certificateproperty 77
requestrdn 78
subjectrdn 80
subjectaltname 81
timenow 83
timenowutc 85
Data Lookup Tags 86
adsquery 86
Text Transformation Tags 92
toupper 92
tolower 93
reversedn 94
iptohexstring 95
uuidtohexstring 96
base64tohexstring 97
ASN.1 Basic Types 98
objectid 98
character string types 99
date string types 100
bitstring 101
octetstring 102
boolean 103
integer 104
enumerated 105
null 106
ASN.1 Constructed Types 107

sequence 107
set 108
explicit 109
ASN.1 Raw Data 110
Unsupported ASN.1 Types 111
Using a custom extension definition to enforce subject name requirements ... 111
Using a custom extension definition to enforce subjectAltName extension requirements 112
Using a custom extension definition to lookup data for the subject DN 113
Allowed X.509 Extensions 115
Adding well known extensions 116
Adding extensions by OID 117
Removing extensions from the list 117

Event Log 118
Events IDs used 119

Regular expressions 120
Syntax 120
Common characters 120
Special characters 120
Non printable characters 120
Characters for pattern expressions 121
Character sets 121
Standard character sets 121
Custom defined character sets 122
Repeat quantifiers 123
Standard quantifiers 123
Reluctant quantifiers 123
Possessive quantifiers 124
Character boundaries 124
Alternative expressions 124
Grouping 124
Options 125
Regular Expression Test Utility 126

Common Problems 127

Reference 128
Tools 128
Literature 128
Links 129

Overview

What it is the keyon / true-Xtender?

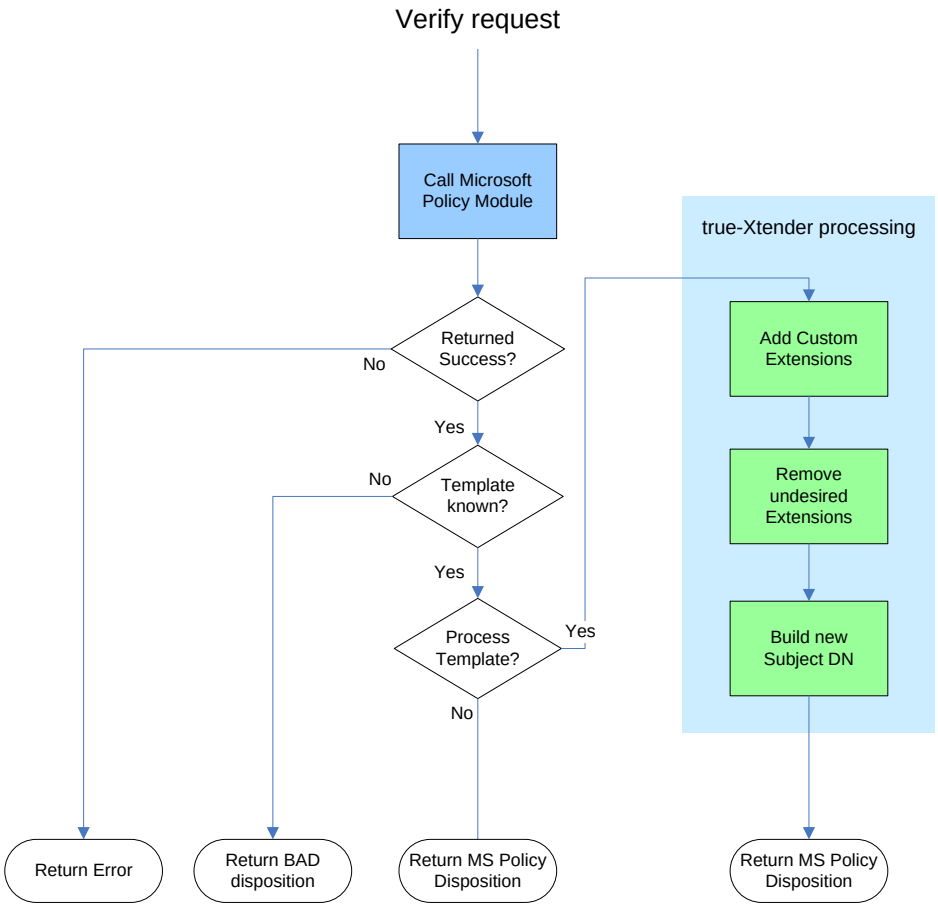
keyon / true-Xtender is a policy module for Certificate Services included in Microsoft Windows Server 2003 that allows to alter the subject DN and X.509 extensions of certificates issued with Certificate Services. It allows certificate content manipulation at a level that is otherwise not possible with Certificate Services alone.

Key Features

- Works with both Standalone CA and Enterprise CA types.
- The functionality can be defined for each certificate template separately.
- Retains the functionality of the original Certificate Services policy module including the special Enterprise functionality.
- The subject distinguished name (SubjectDN) of the issued certificate can be set at will. true-Xtender allows e.g. to use the common name from the request but add static RDN elements for the directory information tree (DIT) that does not need to be available in the Active Directory. This functionality also allows correcting the subject DN specified in PKCS#10 requests generated with 3rd party software.
- Remove unwanted X.509 certificate extensions the Certificate Services automatically adds to the certificate. This is especially useful when certificates are issued for non-Microsoft clients that are probably confused by the additional extensions.
- Add custom extensions that can be defined using an XML based description language. Custom extensions can contain dynamic data taken from the request and provide full control over the X.509 extension content. The XML description language supports all ASN.1 constructs that are commonly used in X.509 extensions.
- The Active Directory schema does not need to be changed for the keyon / true-Xtender.
- Activities are reported in the Windows Event Log.

Request processing with true-Xtender

The request flow when true-Xtender is installed and Certificate Services processes a certificate request is as follows:



1. The delegate policy module is called (usually the original Microsoft Policy Module) to process the request. If the request fails of the template must not be processed by true-Xtender, the request processing stops.

2. Custom X.509 extensions are optionally added according to the configuration for the template used.

3. Unwanted X.509 extensions are optionally removed according to the configuration for the template used.

4. The subject DN is optionally altered according to the configuration for the template used.

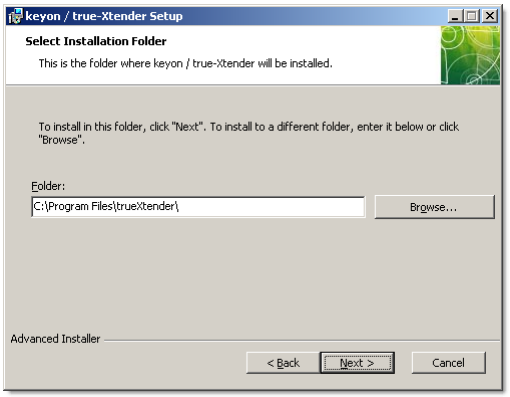
Installation

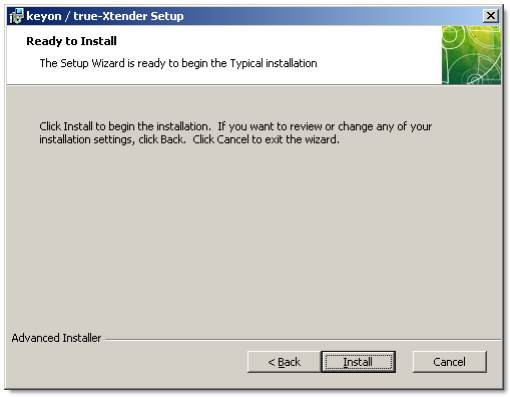
keyon / true-Xtender is shipped as a Windows Installer (MSI) package. Simply double click the installation file provided:

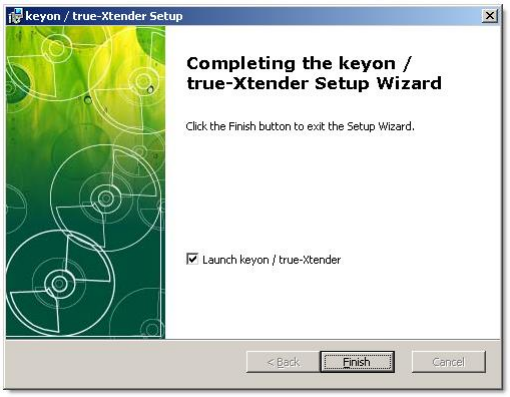
If you are installing the true-Xtender on a 64-Bit Microsoft Server System you must install the 64-Bit true-Xtender installation package. The 32-Bit true-Xtender cannot be used on a 64-Bit system.

The installation is the started and shows mainly the following four screens:

1.
keyon / true-Xtender Setup
Welcome to keyon / true-Xtender Setup Wizard
The Setup Wizard will install keyon / true-Xtender on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
< Back
Next >
Cancel

2. 
keyon / true-Xtender Setup
Select Installation Folder
This is the folder where keyon / true-Xtender will be installed.
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".
Folder:
C:\Program Files\trueXtender\
Browse...
Advanced Installer
< Back
Next >
Cancel

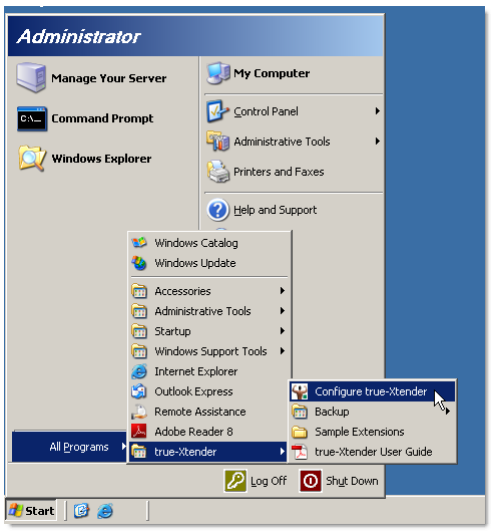
3. 
keyon / true-Xtender Setup
Ready to Install
The Setup Wizard is ready to begin the Typical installation
Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
Advanced Installer
< Back
Install
Cancel

4. 
keyon / true-Xtender Setup
Completing the keyon / true-Xtender Setup Wizard
Click the Finish button to exit the Setup Wizard.
Launch keyon / true-Xtender
< Back
Finish
Cancel

You can use *Add or Remove Programs* in the Windows control panel to repair or remove the keyon / true-Xtender installation.

After successfully installing the keyon / true-Xtender a new entry in the start menu is created under

Start → All Programs → true-Xtender

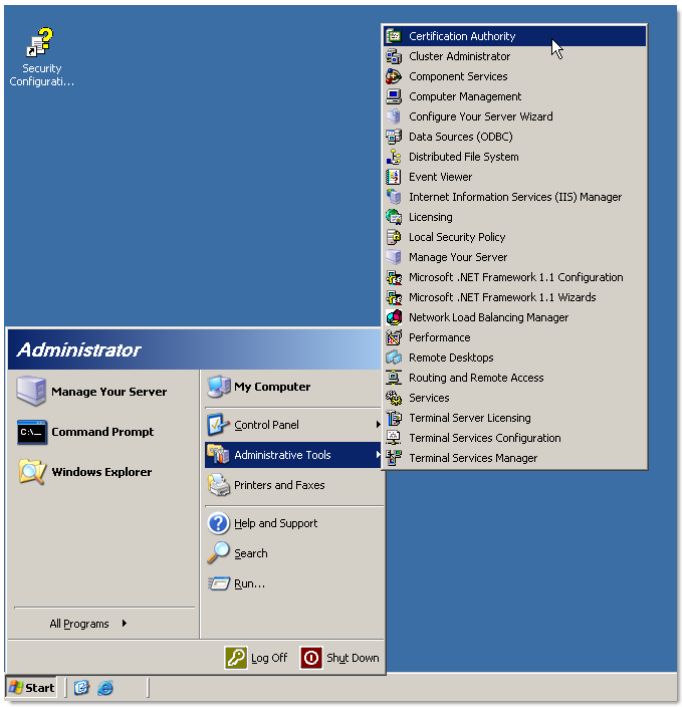


You are now ready to configure Microsoft Certificate Service the use the keyon / true-Xtender as the policy module to use.

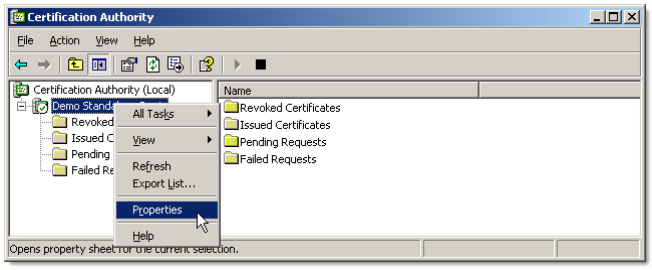
Configure Certificate Services for keyon / true-Xtender

Start the Certificate Services Management console by selecting

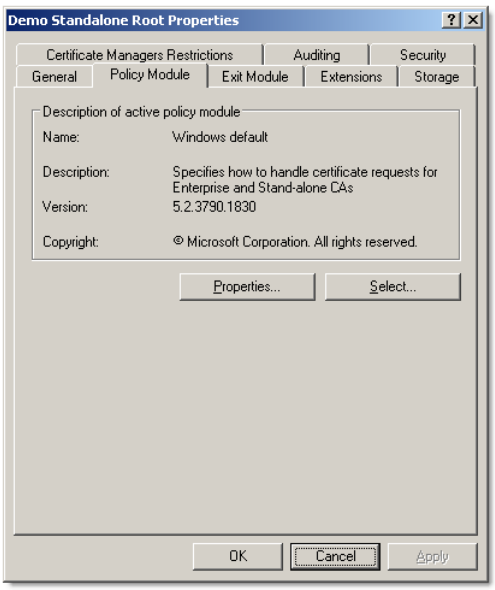
Start → Administrative Tools → Certificate Authority



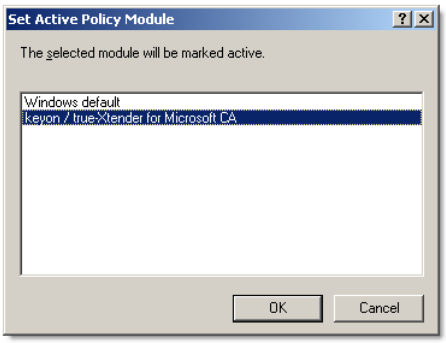
Select *Properties* in the context menu of the CA:



Select the *Policy Module* tab in the properties dialog shown to view the currently active policy module:

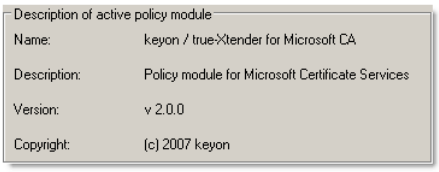


Click *Select...* to change the policy module. The popup dialog will now list the keyon / true-Xtender policy module:

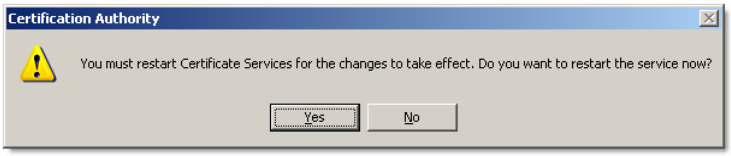


Select the keyon / true-Xtender policy module from the list and click *OK*.

The description of the active policy module will now show the keyon / true-Xtender properties:

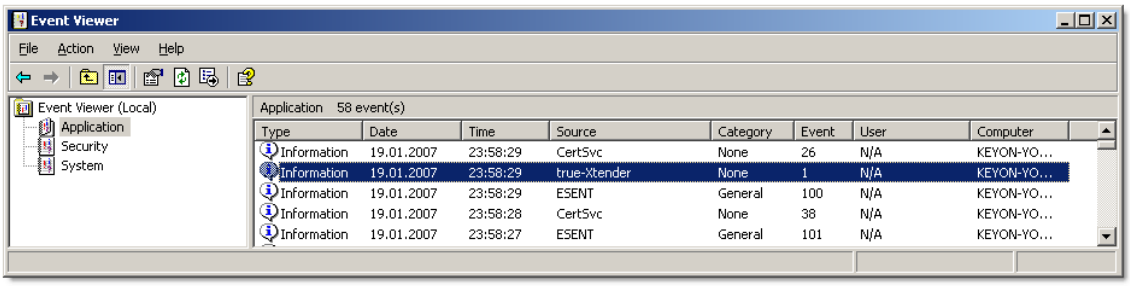

Description of active policy module
Name: keyon / true-Xtender for Microsoft CA
Description: Policy module for Microsoft Certificate Services
Version: v 2.0.0
Copyright: (c) 2007 keyon

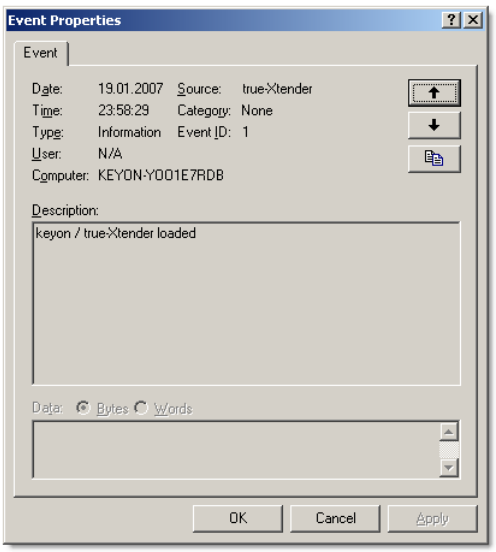
After clicking *OK* or *Apply* in the CA properties dialog the new policy module will be used. If Certificate Services is currently running, the following dialog will be shown:

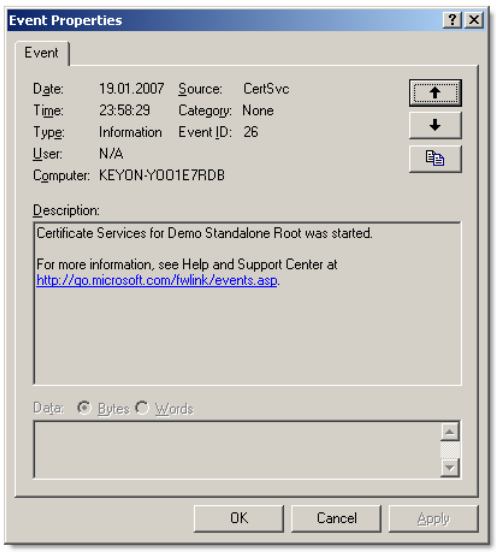

Certification Authority
You must restart Certificate Services for the changes to take effect. Do you want to restart the service now?
Yes
No

Click *Yes* to restart Certificate Services and activate the keyon / true-Xtender policy module.

The Application Event Log will now show two entries whenever Certificate Services are started:


Event Viewer
File Action View Help
Event Viewer (Local)
Application
Security
System
Application 58 event(s)
Type Date Time Source Category Event User Computer
Information 19.01.2007 23:58:29 CertSvc None 26 N/A KEYON-YO...
Information 19.01.2007 23:58:29 true-Xtender None 1 N/A KEYON-YO...
Information 19.01.2007 23:58:29 ESENT General 100 N/A KEYON-YO...
Information 19.01.2007 23:58:28 CertSvc None 38 N/A KEYON-YO...
Information 19.01.2007 23:58:27 ESENT General 101 N/A KEYON-YO...





Backup and restore

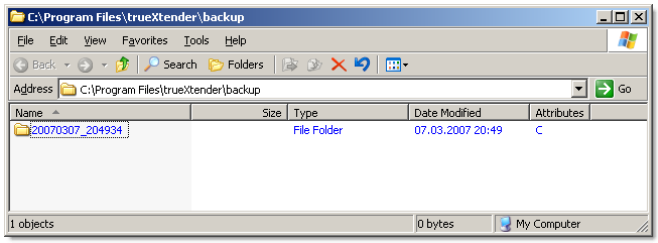
Creating a backup of the current true-Xtender configuration

The true-Xtender start menu contains a sub menu *Backup*:

Selecting *Backup true-Xtender configuration* from the *Backup* sub menu will create a backup of the current true-Xtender configuration as well as all custom extension files configured:

Note that the backup procedure is silent, i.e. no confirmation is shown after the backup is created. This allows creating regular backups using a scheduled job.

Each backup is stored in a separate sub folder using the current date and time as the folder name:

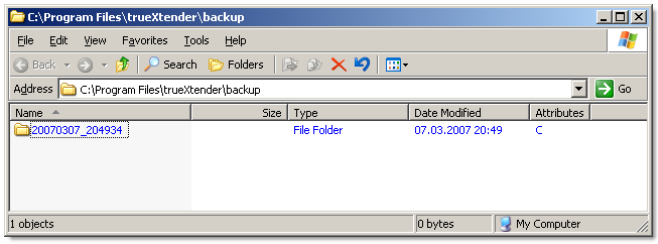


Each backup folder contains a backup of the true-Xtender registry settings as well as copies of all configured custom extension definition files.

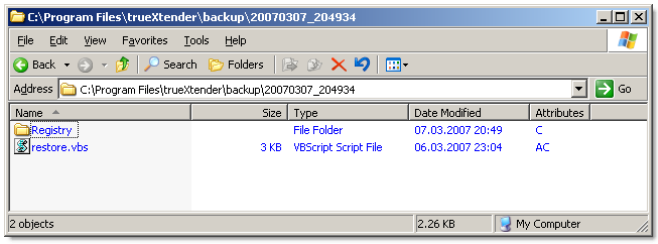
Backup folders are not removed if you uninstall the keyon / true-Xtender.

Restoring a true-Xtender configuration from a backup

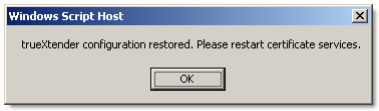
In the true-Xtender start menu, select *Backup* Configuration backups to open the backup root folder:



Select the backup you'd like to restore by double clicking the folder:



Restore the selected backup by double clicking the *restore.vbs* script. After the backup is restored, the following dialog is shown to remind you to restart Microsoft Certificate Services to activate the restored configuration.



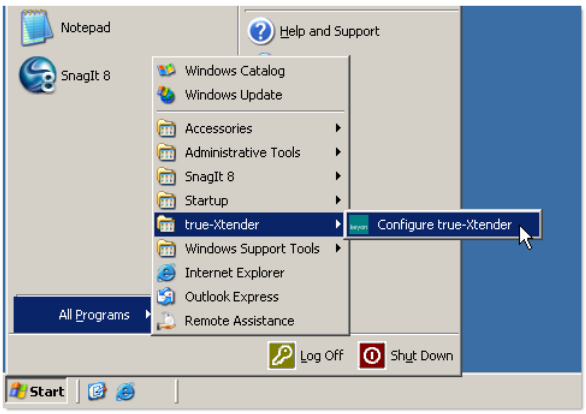
Restoring a backup will overwrite the current true-Xtender configuration and restore all custom extension files that are contained in the backup. Note that the custom extension files are restored to their original location thereby overwriting any present files.

Configuration

Using true-Xtender configuration application

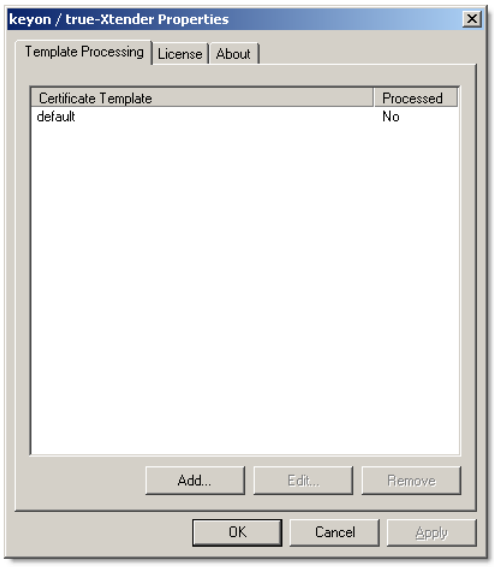
Start the true-Xtender configuration by selecting

Start → All Programs → true-Xtender → Configure true-Xtender

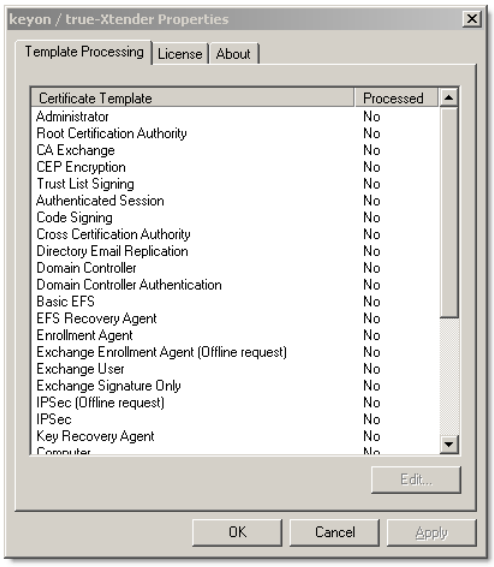


The true-Xtender configuration will be opened:

Standalone CA



Enterprise CA



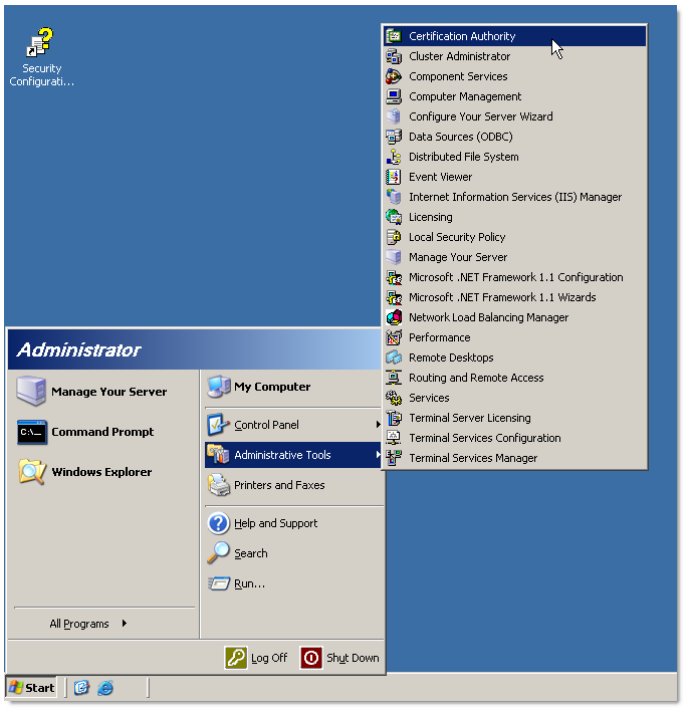
You cannot configure the underlying Microsoft policy module when true-Xtender is configured using the provided configuration application.

Configure true-Xtender from within the Certificate Services management console if you want to configure the underlying Microsoft policy module.

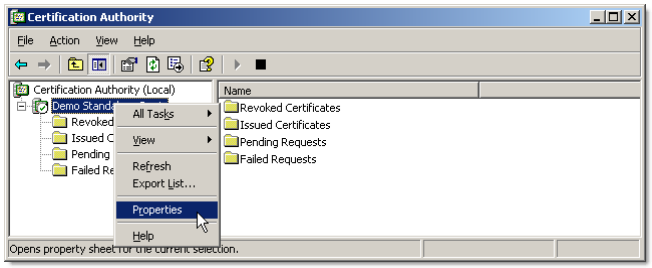
Configure within Certificate Services Management

Start the Certificate Services Management console by selecting

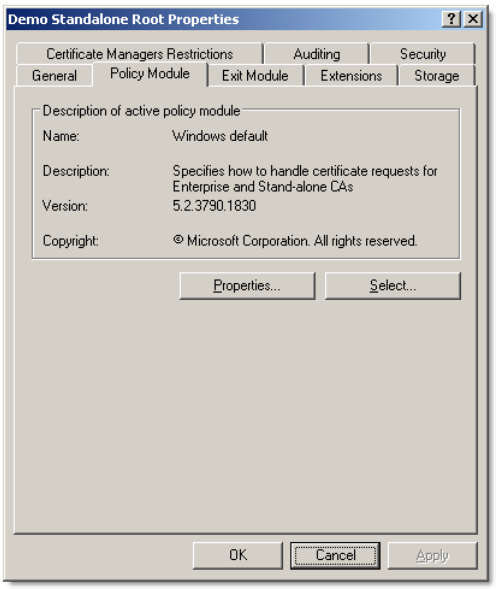
Start → Administrative Tools → Certificate Authority



Select *Properties* in the context menu of the CA:

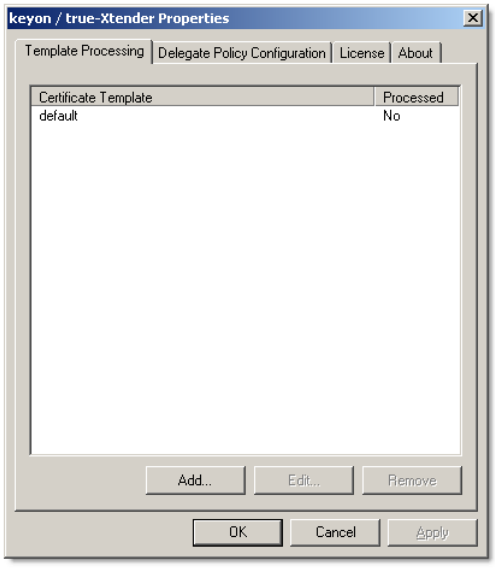


Select the *Policy Module* tab in the properties dialog shown to view the currently active policy module:

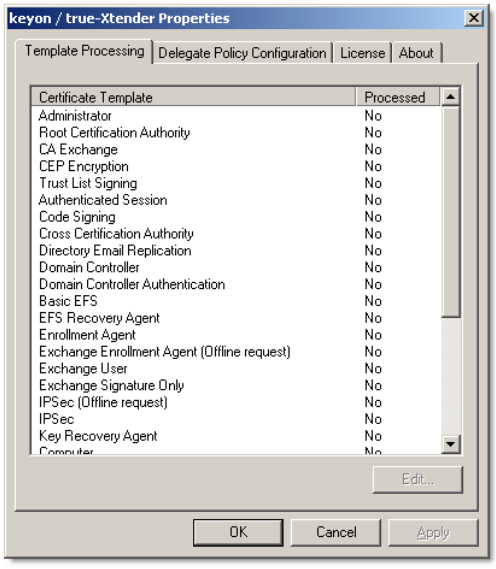


Select *Properties...* to open the true-Xtender configuration:

Standalone CA

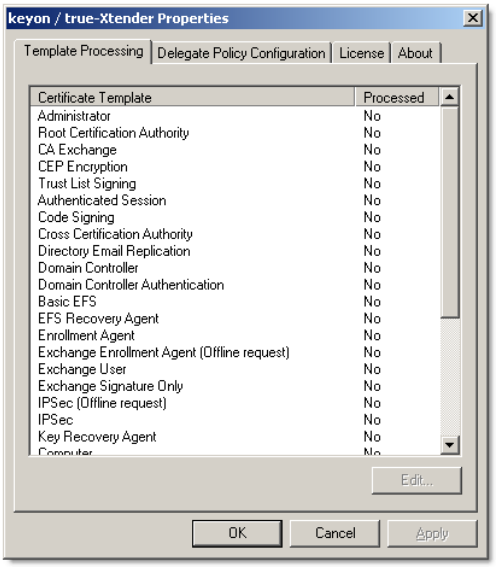


Enterprise CA



Template Processing

The *Template Processing* tab shows the available certificate templates and if a specific certificate template is configured for processing by true-Xtender.



The actual template processing page contents and options depend if an Enterprise CA or a Standalone AC is used.

You can open the dialog to configure a specific template by selecting the entry and click the *Edit...* button or by double clicking the certificate template name.

Enterprise CA Types

true-Xtender will always show all certificate templates found in the Active Directory, even if the CA supports only a subset of the certificate templates:

The dialog shows the template display names.

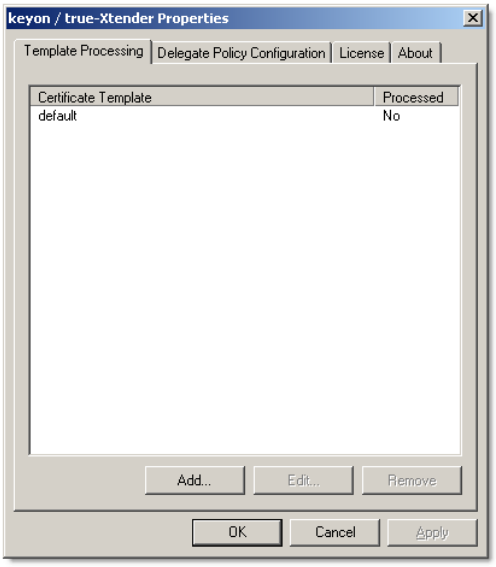
Standalone CA Types

Certificate Services for Standalone CAs do not support certificate templates on their own. However true-Xtender provides a means to add templates even for standalone CAs.

Managing standalone templates

The default template

The keyon true-Xtender automatically adds a default template with name *default* when used with a standalone CA type. By default, the true-Xtender processing for the *default* template is deactivated however.

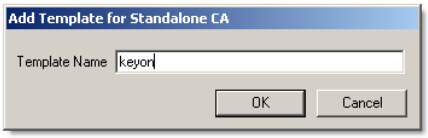


The default template is used for all certificate requests for which no certificate template is specified. See below to learn how to specify a certificate template when using a standalone CA type.

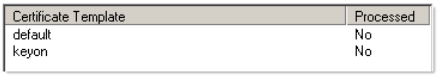
You cannot delete the default template.

Adding custom templates

On the *Template Processing* tab, click the *Add...* button. Enter the name of the new template in the popup dialog that is shown:



After clicking ok, the new template will be added to the templates list:



Template names are case sensitive. However you cannot add two templates which differ only by case.

You should not use the space character or any other special character in the template name.

Removing custom templates

On the *Template Processing* tab, select the template you'd like to remove and click the *Remove* button.

Specifying the template to use for a request with a Standalone CA

You can set the certificate template to use with a standalone CA either when you create the request or when the request is pending.

New Requests

If new requests are created with the `certreq` utility you can set the template to use by adding the following lines to the policy INI file:

```
[RequestAttributes]
CertificateTemplate = "keyon"
```

If you set the certificate template in the request attribute as described above, Certificate Services will include an X.509 extension with the certificate template name and extension OID `1.3.6.1.4.1.311.20.2`.

You can use true-Xtender to filter out this possibly unwanted extension.

Pending Requests

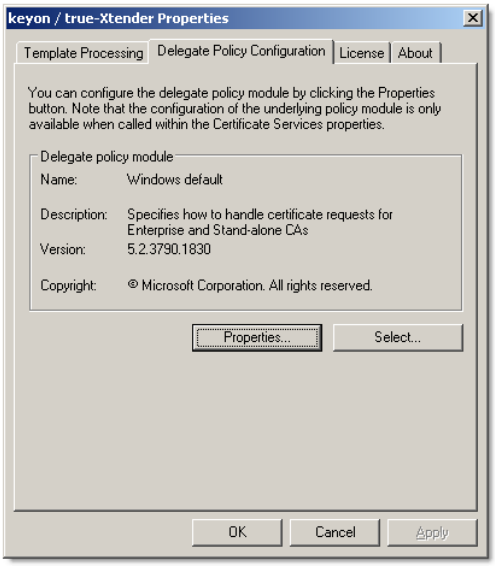
You can set the template to use for pending requests with the `certutil` utility:

```
certutil -setattributes id "StandaloneCertificateTemplate:keyon"
```

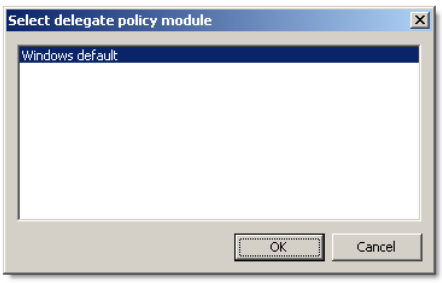
You cannot set the *CertificateTemplate* request attribute using `certutil` when you have a standalone CA. You must use the request attribute StandaloneCertificateTemplate instead to set the template to use for the keyon / true-Xtender.

Delegate Policy Configuration

The *Delegate Policy Configuration* tab allows selecting and configuring the delegate policy module to use by the true-Xtender during the request processing:

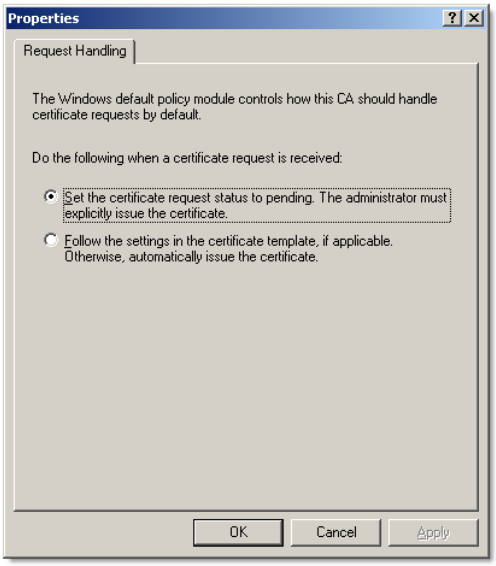

keyon / true-Xtender Properties
Template Processing
Delegate Policy Configuration
License
About
You can configure the delegate policy module by clicking the Properties button. Note that the configuration of the underlying policy module is only available when called within the Certificate Services properties.
Delegate policy module
Name: Windows default
Description: Specifies how to handle certificate requests for Enterprise and Stand-alone CAs
Version: 5.2.3790.1830
Copyright: © Microsoft Corporation. All rights reserved.
Properties...
Select...
OK
Cancel
Apply

After clicking *Select...*, the available policy modules on the system are listed and you can select the delegate policy module to use for the request processing:



Note that searching the available policy modules may take while. The true-Xtender policy module is never shown in the list as it is not a valid selection for the delegate policy module.

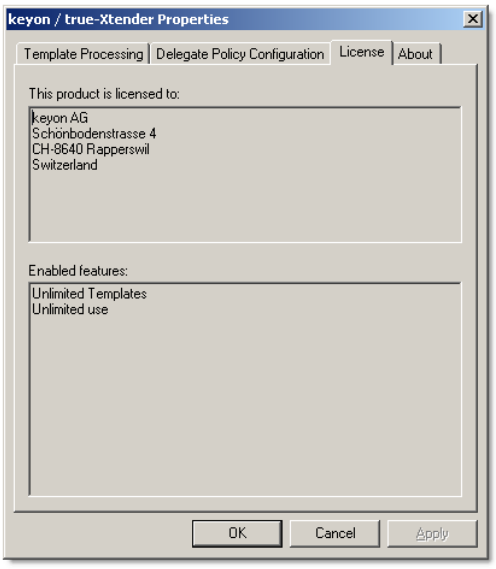
You can edit the configuration of the delegate policy module by clicking *Properties...*:


Properties
Request Handling
The Windows default policy module controls how this CA should handle certificate requests by default.
Do the following when a certificate request is received:
Set the certificate request status to pending. The administrator must explicitly issue the certificate.
Follow the settings in the certificate template, if applicable. Otherwise, automatically issue the certificate.
OK
Cancel
Apply

Consult the Microsoft documentation for details on the Microsoft policy configuration.

License

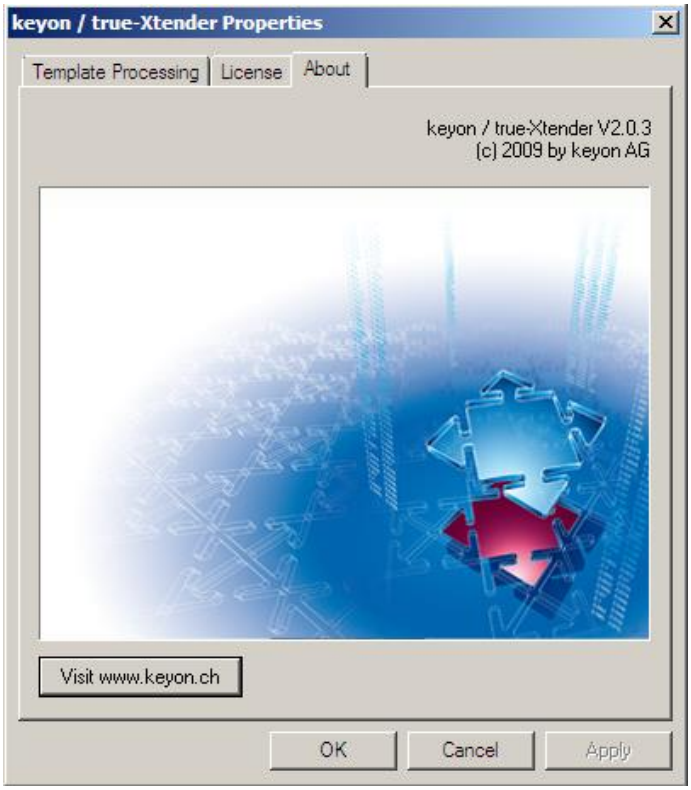
The *License* tab shows the true-Xtender license along with possible restrictions:



You must make sure to install and use the keyon / true-Xtender only in accordance with the license agreement.

About

The *About* tab shows the true-Xtender version and provides a button to open a web browser and open the keyon web site:

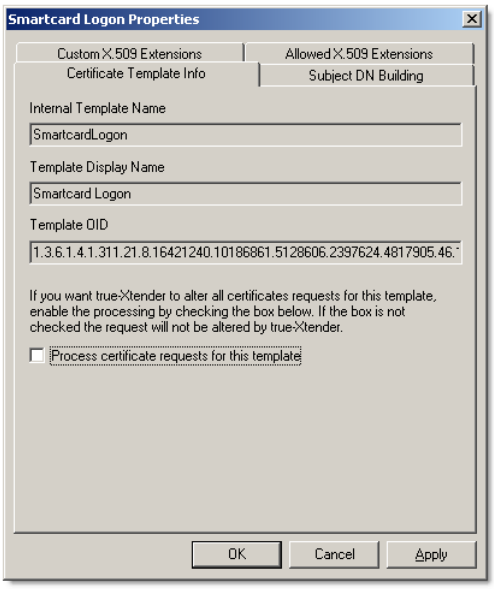


Template Configuration

A specific template is configured by selecting the template in the *Template Processing* tab and clicking the *Edit...* button or double clicking the template.

Certificate Template Info

The *Certificate Template Info* tab shows the template names and allows enabling or disabling the processing by the keyon / true-Xtender for this certificate template:


Smartcard Logon Properties
Custom X.509 Extensions
Allowed X.509 Extensions
Certificate Template Info
Subject DN Building
Internal Template Name
SmartcardLogon
Template Display Name
Smartcard Logon
Template OID
1.3.6.1.4.1.311.21.8.16421240.10186861.5128606.2397624.4817905.46.
If you want true-Xtender to alter all certificates requests for this template, enable the processing by checking the box below. If the box is not checked the request will not be altered by true-Xtender.
Process certificate requests for this template
OK
Cancel
Apply

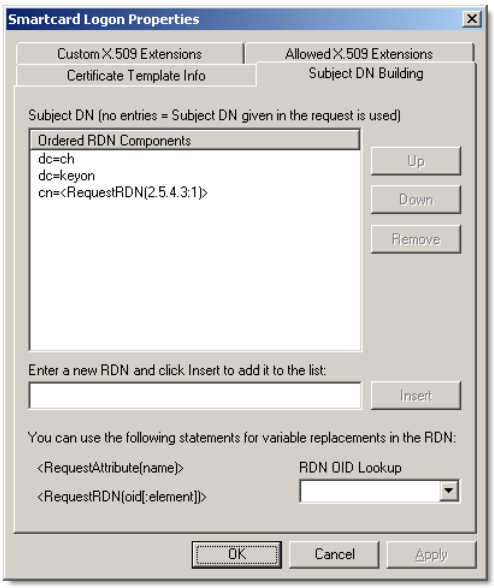
The *Internal Template Name* is the certificate template specification to use when creating the certificate request while the *Template Display Name* is the friendly name shown in the Certificate Services management console.

Enabling or disabling processing of this template

Select *Process certificate requests for this template* if you want to modify the issued certificates with the keyon / true-Xtender. If this checkbox is not selected, certificate requests for the certificate template are not processed by true-Xtender.

Subject DN Building

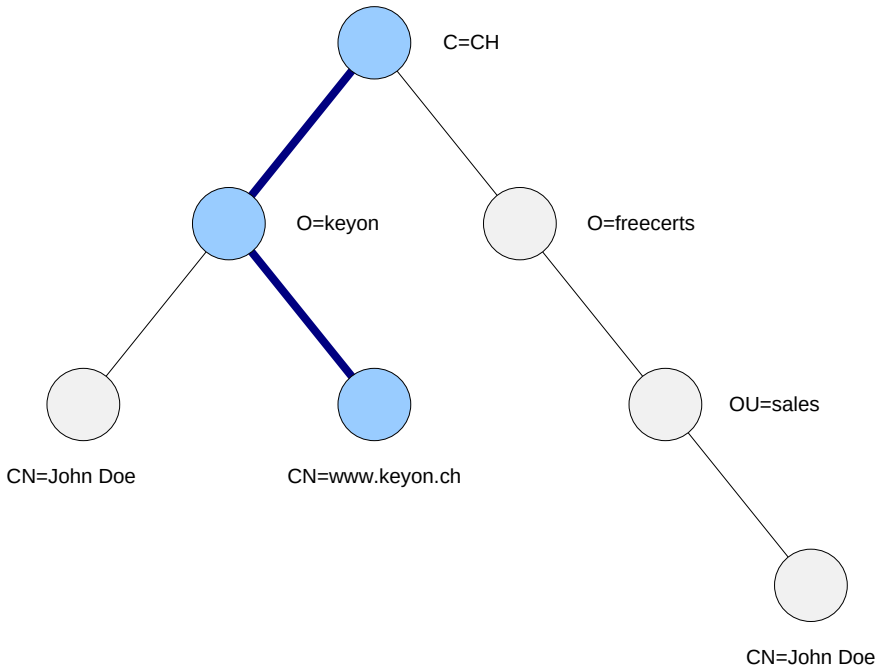
The *Subject DN Building* tab allows customizing the DN of the issued certificate. The resulting DN may use relative distinguished name (RDN) parts of the original request, request attributes and fixed parts to build the intended subject RDN elements.



If you do not specify any RDN elements, the subject DN from the original request will be used.

Directory Information Tree, RDNs and Distinguished Names

The directory information tree (DIT) provides a way to refer to the data stored in a X.500 or LDAP directory.



A distinguished name (DN) is the string representation of an entry's name and location in a directory. A DN describes a path to exactly one directory entry and is made up of a number of components called relative distinguished names (RDNs). Each RDN identifies a specific entry in the directory.

The highlighted path in the DIT shown above is described by the following DN which is made up of the three RDNs C, O and CN.

```
C=CH, O=keyon, CN=www.keyon.ch
```

Note that there a two naming schemas in use: The standard X.500 schema which uses countries or organizations as root elements and the DNS schema which uses domain components as the root elements.

In a certificate, the DN `C=CH, O=keyon, CN=www.keyon.ch` is encoded in ASN.1 as follows:

```
SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER countryName (2 5 4 6)
      PrintableString 'CH'
      }
    }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER organizationName (2 5 4 10)
      PrintableString 'keyon'
      }
    }
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER commonName (2 5 4 3)
      PrintableString 'www.keyon.ch'
      }
    }
  }
```

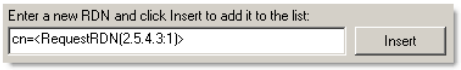
DNs are usually printed in reverse order as the last element defines the interesting end entity. Printing the RDNs in reverse order thus improves the readability:

```
cn=www.keyon.ch o=keyon, c=CH
```

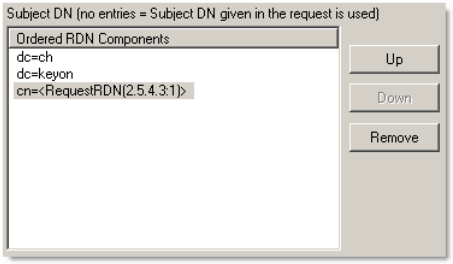
In the certificate structure itself however, the order should be hierarchical top down, starting with the root node (c=ch) and ending with the end entity node.

Adding a RDN element to the subject DN

Add the RDN into the edit field under *Enter a new RDN and click Insert to add it to the list*:



See below for the special syntax used to add request attribute contents or contents of the original request DN. After clicking the *Insert* button, the new element is inserted in the list of RDNs:



Empty elements are not added to the subject DN. If variables are used and the value is empty (i.e. after the expansion the element reads something like cn=), the element will not be included.

You can use the custom extension function with a not included empty extension that just checks that required elements are present.

Using static elements for RDN elements

Static elements are simply entered like:

```
dc=keyon
dc=keyon, dc=ch
```

If the static text contains one of the following special characters

```
, + = " < > # ;
```

the whole RDN value (expression right to =) must be quoted using the " character. If the RDN value itself contains a " character it must be double quoted ("").

Samples

To insert the organization name *keyon, Inc.*, you must enter it as:

```
o="keyon, Inc."
```

To insert the organization name *keyon "1a" Inc.*, you must enter it as:

```
o="keyon ""1a"" Inc."
```

Using the content of request attributes for RDN elements

The following syntax is used to insert the content of a request attribute:

```
<RequestAttribute(name)>
```

Where *name* is the case sensitive name of the request attribute whose contents are to be inserted. Note that the request attribute may also contain the RDN type string, e.g. `cn=`.

You can provide the complete DN in a request attribute. Simply leave out the static RDN prefix.

You can add or change request attributes within the keyon / true-Xtender processing scope in a custom extension by using the `<definerequestattribute>` tag.

Samples

```
cn="<RequestAttribute(firstname)> <RequestAttribute(surname)>"
```

If the request attribute *firstname* contains *John* and the request attribute *surname* contains *Doe*, the RDN created would be *cn=John Doe*. The use of `"` ensures that names containing special characters (`, + = " < > # ;`) are inserted correctly. The surrounding quotes are not inserted into the resulting RDN, it is thus always safe to use quotes.

```
<RequestAttribute(newdn)>
```

If the request attribute *newdn* contains cn=*John Doe, dc=keyon, dc=ch,* the DN created would be cn=*John Doe, dc=keyon, dc=ch* if no other RDN elements are configured. However you must make sure that the DN provided in the attribute uses appropriate quoting if some RDN elements contain special characters.

Using the content of request RDNs

The following syntax is used to insert the content of a RDN from the original request:

```
<requestrdn(oid[:element])>
```

Where *oid* is the OID of the RDN whose contents are to be inserted and *element* is the number of the RDN to use if multiple RDNs with the same OID are present. If *element* is not given, the first RDN with the given OID is used.

Elements taken from the original request containing " characters are automatically expanded, i.e. each " character is replaced with "". You must make sure to quote the whole RDN value to support original request values containing special characters (e.g. `CN="<RequestRDN(2.5.4.3)>"`).

Samples

Consider the following original request DN:

```
cn=John Doe, cn=Users, dc=keyon, dc=ch
```

The contents of the RDN elements can be retrieved as follows:

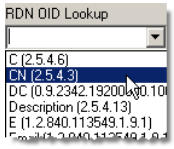
```
<RequestRDN(2.5.4.3)>                       → John Doe
<RequestRDN(2.5.4.3:1)>                     → John Doe
<RequestRDN(2.5.4.3:2)>                     → Users
<RequestRDN(0.9.2342.19200300.100.1.25)>    → keyon
<RequestRDN(0.9.2342.19200300.100.1.25:1)>  → keyon
<RequestRDN(0.9.2342.19200300.100.1.25:2)>  → ch
```

OIDs of common RDN elements

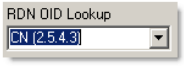
Long name	Short name	OID
CommonName	CN	2.5.4.3
Country	C	2.5.4.6
DeviceSerialNumber	-	2.5.4.5
DomainComponent	DC	0.9.2342.19200300.100.1.25
Email	E	1.2.840.113549.1.9.1
GivenName	G	2.5.4.42
Initials	I	2.5.4.43
Locality	L	2.5.4.7
Organization	O	2.5.4.10
OrganizationalUnit	OU	2.5.4.11
State	ST, S	2.5.4.8
StreetAddress	Street	2.5.4.9
SurName	SN	2.5.4.4
Title	T	2.5.4.12

RDN OID Lookup Utility

The *Subject DN Building* page features a simple RDN OID lookup utility in the lower left corner. To select an RDN, click on the drop list button :



Select the requested RDN from the drop down list and it will be entered into the edit field:



You can now select the OID part and use *Ctrl-C* or *right click / Copy* to copy the OID to the clipboard.

Removing a RDN element

In order to remove a RDN element, select the element and click the *Remove* button.

Edit a RDN element

You cannot currently edit RDN elements in the list. If you like to change a RDN element, remove the RDN element first and add the changed RDN element again.

Reorder RDN elements

To reorder the RDN elements, select the RDN element you'd like to move and use the *Up* and *Down* buttons to change the position in the RDN order.

The order of RDNs is not completely defined by this dialog but by a registry setting of Certificate Services which affects all certificates issued as well. The resulting subject DN will by default always use the following hierarchical order for RDN elements:

```
Country
DomainComponent
State
Locality
Organization
OrganizationalUnit
CommonName
Email
```

The order within one hierarchy element (e.g. DC) can be defined with the keyon / true-Xtender however. Note that other orders of the hierarchy would not make sense in most of the cases and would certainly not be X.500 compliant.

By editing the following registry entry it is possible to change this order:

```
HKEY_LOCAL_MACHINE
   \SYSTEM
      \CurrentControlSet
         \Services
            \CertSvc
               \Configuration
                  \CA Name → SubjectTemplate
```

Samples

The following samples show how a request DN is altered depending on the configuration.

Keep request DN

Challenge: The original DN from the request should be used for the issued certificate.

Request DN
`cn=John Doe, cn=Users, dc=keyon, dc=ch`

Request attributes
-

Ordered RDN Components
-

Resulting Subject DN
`cn=John Doe, cn=Users, dc=keyon, dc=ch`

Using original request RDN element

Challenge: The first CN element from the request DN must be used and the rest of the DN must be hard coded.

Request DN
`cn=John Doe, cn=Users, dc=keyon, dc=ch`

Request attributes
-

Request attributes

```
c=CH
o=keyon
ou=Engineering
cn="<RequestRDN(2.5.4.3)>"
```

Resulting Subject DN

```
cn=John Doe, ou=Engineering, o=keyon, o=CH
```

Use request attributes

Challenge: The CN and the Email elements in the subject DN must be taken from request attributes and the rest of the DN must be hard coded.

Request DN

```
cn=John Doe, cn=Users, dc=keyon, dc=ch
```

Request attributes

```
mycn:Jane Sample
myemail:jane.sample@keyon.ch
```

Ordered RDN Components

```
dc=ch
dc=keyon
cn="<RequestAttribute(mycn)>"
email="<RequestAttribute(myemail)>"
```

Resulting Subject DN

```
email=jane.sample@keyon.ch, cn=Jane Sample, dc=keyon, dc=ch
```

Externally build DN

Challenge: The complete subject DN to use must be provided in a request attribute.

Request DN

```
cn=Jane Sample, cn=Users, dc=keyon, dc=ch
```

Request attributes

```
mydn:cn=John Doe, o=keyon, c=CH
```

Ordered RDN Components

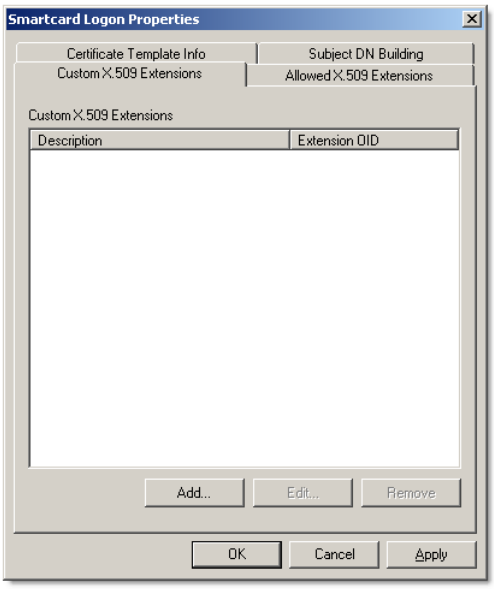
```
<RequestAttribute(mydn)>
```

Resulting Subject DN

```
cn=John Doe, o=keyon, c=CH
```

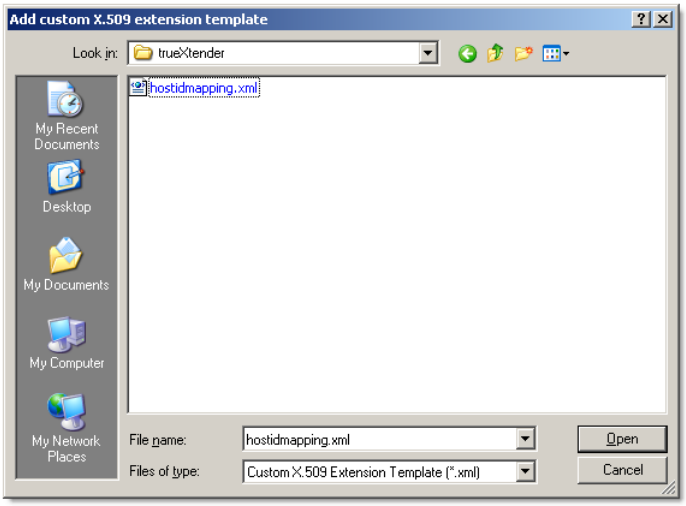
Custom X.509 Extensions

The *Custom X.509 Extensions* tab allows adding custom extensions to certificates.



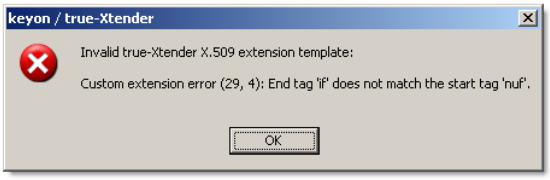
Adding custom extensions

Click on the *Add...* button to select the XML file containing the custom extension description. See *Custom extensions definition language* for a description of the XML contents.



The extension is validated and if found ok, the extension is added to the list of custom extensions:

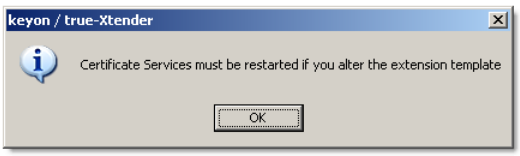
In case of an error in the XML description of the custom extension, the kind of error and if possible the location in the input file (line, column) is shown:


keyon / true-Xtender
Invalid true-Xtender X.509 extension template:
Custom extension error (29, 4): End tag 'if' does not match the start tag 'nuf'.
OK

Correct the problem and try to add the extension again.

Edit custom extension files

Select the extension and click the *Edit* button to start the text editor with the custom extension file opened. If Certificate Services is running, the following message will be shown to remind you that custom extensions are only loaded at startup of Certificate Services:



If the custom extension file is not valid after editing, Certificate Services will not start. Check the Event Log for details on the error.

In order to check the file in the configuration dialog you can remove it and add it again.

Removing custom extensions

Simply select the extension and click the *Remove* button to remove a custom extension.

The external file with the custom extension is not removed and can be added again.

Custom extensions definition language

The custom extension insertion mechanism is a very powerful feature of the keyon / true-Xtender that allows describing X.509 certificate extensions using a XML based description. Since the description allows inserting dynamic content like request attributes or request subject DN elements, even extensions that contain data specific to the certificate holder like e.g. logon id can be easily created without the need to implement additional code.

A few sample extensions are installed with the keyon / true-Xtender and are available through a link in the keyon / true-Xtender start menu entry.

The creation of custom extensions requires knowledge of ASN.1 types and structures. While true-Xtender checks the custom extension XML file for syntax and ensures that only valid DER encoded ASN.1 structures are created (unless raw ASN.1 data is inserted), it cannot not ensure that the ASN.1 structures comply with the actual definition of a specific extension.

If not defined properly, the generated extension may not be parsed by clients correctly which may ultimately result in a crash of an application that uses the certificate.

See the reference sections for literature and tools that will help you when you are going to create custom extensions.

You can also overwrite any standard extension included by the Certificate Services policy module using the custom extensions mechanism. If you want to add e.g. a Certificate Policies extension that is different for a specific certificate type, simply create a custom extension with the certificate policies OID and define the Certificate Policies extension using the XML description language.

Illustrated custom extension example

As an example, we want to include the non-critical IBM *hostIdMapping* extension if the certificate request has the attribute `hostid` set. If the request attribute is not set, we do not want to include the extension. This example will show the use of most available tag groups that are provided for custom extensions.

The *hostIdMapping* extension (OID 1 3 18 0 2 18 1) is an IBM extension, also available for public use. RACF automatically maps a valid certificate to the RACF user ID provided in the extension. The ASN.1 definition taken from the IBM documentation is as follows:

```
id-ce-hostIdMapping OBJECT IDENTIFIER ::= { 1 3 18 0 2 18 1 }

HostIdMapping ::= SEQUENCE {
  hostName [1] IMPLICIT IA5String,
  subjectId [2] IMPLICIT IA5String
}
```

To achieve our goal, the following XML definition for the extension is created:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE extension SYSTEM "trueXtender_2.3.3.dtd">

<extension oid="1.3.18.0.2.18.1" critical="false"
           onempty="remove" name="IBM HostIdMapping">

  <define name="hostid">
     <requestattribute name="hostid"/>
  </define>

  <!-- only add if request attribute is present -->
  <if name="hostid">
       <sequence>
            <ia5string tag="1">
                 SAMPLE_HOST_NAME
            </ia5string>
            <ia5string tag="2">
                 <tolower>
                      <insert name="hostid"/>
                 </tolower>
            </ia5string>
       </sequence>
  </if>

</extension>
```

The following gives a description of the tags used in the example:

The tag names are not case sensitive. However if DTD validation is used when authoring the XML file, the tag names must be in lower case.

XML header (optional)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

The XML version (always 1.0) and the character set encoding are specified. Note that the header is optional but its use is recommended.

DOCTYPE Declaration (optional)

```
<!DOCTYPE extension SYSTEM "trueXtender_2.3.3.dtd">
```

DTD validation is not performed by true-Xtender when loading the XML file. However adding the DOCTYPE declaration along with the DTD file may help authoring extensions with editors that use the DTD for syntax validation.

XML comments

```
<!-- only add if request attribute is present -->
```

XML comments can be used to include descriptions or to comment out elements.

Extension definition

```
<extension oid="1.3.18.0.2.18.1" critical="false"
           onempty="remove" name="IBM HostIdMapping">
  ...
</extension>
```

The `extension` tag defines the OID of the extension, the criticality, the display name in the true-Xtender GUI and defines if the action if the extension does not have any content.

Every custom extension must have an `extension` tag as the root tag. Each custom extension XML file can have only one `extension` tag.

Variable definition

```
<define name="hostid">
   ...
</define>
```

The `define` tag allows to define variables for use e.g. in logic tags. The `define` tag assigns the enclosed content to the named variable, in this case `hostid`.

Request attribute access

```
<requestattribute name="hostid"/>
```

The `requestattribute` tag inserts the content of the named request attribute as text. If the request attribute is not set, an empty string is inserted.

Logic tag

```
<if name="hostid">
   ...
</if>
```

The `if` tag adds the included content only if the condition is matched. In this case the enclosed content is added only if the variable `hostid` is not empty. If the hosted variable is empty, i.e. the request attribute `hostid` is not set, nothing will be added and we have an empty extension. Since the extension tag requests that empty extensions are not added, the keyon / true-Xtender will add the extension only if the request attribute `hostid` is set.

ASN.1 SEQUENCE tag

```
<sequence>
   ...
</sequence>
```

The `sequence` tag will generate an ASN.1 sequence containing the enclosed ASN.1 objects.

ASN.1 IA5String tag

```
<ia5string tag="1">
   SAMPLE_HOST_NAME
</ia5string>
```

The `ia5string` tag will encode the contents as an ASN.1 IA5String. In this case, the fixed text SAMPLE_HOST_NAME is encoded. As the ia5string tag has the tag parameter set, the result will be implicitly tagged with value 1 as requested by the extension specification.

Not that leading and trailing whitespace for text content is stripped.

Content transformation tag

```
<tolower>
      ...
</tolower>
```

The `tolower` tag will lowercase the enclosed text. This ensures, that even if the `hostid` was specified mixed case in the request attribute, we will always include a lowercase `hostid` in the extension.

Variable inclusion

```
<insert name="hostid"/>
```

The `include` tag inserts the content of the named variable.

Root Tag

extension

Tag

Tag name	Enclosed content
`extension`	Any number of logic and definition tags but at most one ASN.1 Tag is allowed. The `extension` tag must be the root tag in the custom certificate extension file and it cannot be nested.

Parameters

Tag parameter	Presence	Description
`oid`	mandatory	The OID of the custom extension.
`name`	mandatory	Display name of the custom extension.
`critical`	optional	If `true`, the extension will be marked critical. If parameter is not included or set to `false`, the extension will be non-critical.
`includeempty`	optional, deprecated	If set to `true`, the extension will be included even if it does not have any contents. If parameter is not present or set to `false`, the extension will not be included in the certificate if it does not contain any ASN.1 elements. The `includeempty` parameter is deprecated. Please use the `onempty` parameter instead which provides more options.
`onempty`	optional	Defines the behavior when the evaluation of the extension contents results in an empty ASN.1 structure: `include` Include the extension with empty content. `remove` Do not include the extension. `original` Insert the extension provided by the original policy module if present. `error` Aborts the request processing with an error. If not present defaults to `remove`.

Samples

Samples

```
<extension oid="1.3.18.0.2.18" critical="false"
           name="IBM HostIdMappings">
     <set>
          <sequence>
               <ia5string tag="1">IBM_HOST_PLATFORM</ia5string>
               <ia5string tag="2">user1</ia5string>
          </sequence>
     </set>
</extension>
```

```
<extension oid="2.5.29.17" critical="false" onempty="original"
           name="SubjectAltName">
     <sequence includeempty="false">
          <define name="dnsName">
               <requestattribute name="dnsName"/>
          </define>
          <if name="dnsName">
               <ia5string tag="2">
                    <insert name="dnsName"/>
               </ia5string>
          </if>
     </sequence>
</extension>
```

Variable Related Tags

define

Tag

Tag name	Enclosed content
`define`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be assigned to the variable. Heading and trailing whitespace is removed. If a regular expression with a capture group `(...)` is specified in the `match` parameter, only the matching part of the expanded text will be assigned to the variable.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable to define.
`match`	optional	The `match` parameter can be used to specify a regular expression with a capture group (i.e. an expression placed in brackets) to extract a part of the expanded text. If no capture group `(...)` is specified in the regular expression, the contents will always evaluate to an empty string.
`matchoptions`	optional	Comma separated modification options for the regular expression evaluation. The most useful option is `IGNORECASE` which will force a case insensitive match of the pattern. This parameter is only used when `match` is given.

If you use regular expressions to extract a part of the content, make sure that the regular expression is designed correctly. You can use the regular expression test utility that ships with true-Xtender to check the results of your regular expression with different input.

See section *Regular expressions* for details about regular expressions.

Samples

Samples

```
<define name="sha1withRSAEncryption">1.2.840.113549.1.1.5</define>
```

```
<define name="upn"> <RequestAttribute name="upn"/> </define>
```

Regular expression sample

Samples

```
<define name="host"
        match="^(.+)\.keyon\.ch$" matchoptions="IGNORECASE">
      <insert name="cn"/>
</define>
```

- If the variable `cn` contains `host1.keyon.ch`, the value `host1` would be assigned to the variable `host` as the expression matches. (Note the capture group `(.+)` in the expression above.)
- If the variable `cn` contains `host1.keyon.com`, an empty string would be assigned to the variable `host` as the expression does not match.

undefine

Tag

Tag name	Enclosed content
`undefine`	Text is ignored. No enclosed ASN.1 tags are allowed. After this tag is processed, the variable is no longer defined.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable to undefine.

Samples

Samples
`<undefine name="upn"/>`

insert

Tag

Tag name	Enclosed content
`insert`	Text is ignored. No enclosed ASN.1 tags are allowed.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable content to insert. If the variable is not defined, nothing will be inserted.

Samples

Samples
`<ia5string> <insert name="sha1withRSAEncryption"/> </ia5string>`

If the variable name is not the reserved name of a XML tag, you can also insert the variable contents using the variable name as an XML tag:

```
<sha1withRSAEncryption/>
```

However the use of `insert` is preferred as it makes clear that a variable is inserted.

inserttoken

Tag

Tag name	Enclosed content
`inserttoken`	Text is ignored. No enclosed ASN.1 tags are allowed. The specified variable is tokenized at the given delimiters and the token with the given index is inserted. Heading and trailing whitespace of the token is removed.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable content to tokenize. If the variable is not defined, nothing will be inserted.
`delimiters`	mandatory	The delimiters to use when tokenizing the variable content. Multiple delimiters including the space character can be specified.
`index`	mandatory	The index of the token to insert.

Samples

Samples

```
<ia5string>
   <inserttoken name="dnsNames" delimiters="," index="2"/>
</ia5string>
```

If the variable *dnsNames* contains the string

```
host1.keyon.ch, host2.keyon.ch, host3.keyon.ch
```

the text `host2.keyon.ch` would be inserted by the statement above.

Note that empty tokens are skipped, i.e. if e.g. `","` is specified as the delimiter, the string `one,,two,,,three` is tokenized into three tokens only: `one` at index 1, `two` at index 2 and `three` at index 3.

Logic Tags

if

Tag

Tag name	Enclosed content
`if`	Only enclosed tags are allowed. The enclosed tags will be added to the custom extension if the specified variable exists, is not empty and optionally has a specific value or matches a regular expression.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable to check.
`value`	optional	If given, the variable content is checked against the `value` parameter. Only if the variable content matches the given value, the enclosed tags are added. If this parameter is not given, the enclosed tags are added if the variable is defined and not empty.
`match`	optional	The `match` parameter can be used instead of the `value` parameter to specify a regular expression instead of a fixed value. The enclosed tags are only added if the variable contents match the regular expression exactly. This parameter is only used when `value` is not given.
`matchoptions`	optional	Comma separated modification options for the regular expression evaluation. The most useful option is `IGNORECASE` which will force a case insensitive match of the pattern. This parameter is only used when `match` is given.

If you use regular expressions make sure that the regular expression is designed correctly.

See section Regular expressions for details about regular expressions.

Samples

Samples

```
<if name="email">
      <sequence>
            <ia5string> <insert name="email"/> </ia5string>
      </sequence>
</if>
```

```
<sequence>
      <if name="includeEmail" value="true">
            <ia5string> <insert name="email"/> </ia5string>
      </if>
</sequence>
```

```
<if name="upn"
    match="^administrator@.*$" matchoptions="IGNORECASE">
      <exception>Invalid UPN: <insert name="upn"/> </exception>
</if>
```

This construct would prevent the use of a UPN for the user *Administrator*.

ifnot

Tag

Tag name	Enclosed content
`ifnot`	Only enclosed tags are allowed. The enclosed tags will be added to the custom extension if the specified variable does not exist, is empty or optionally does not have a specific value.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable to check.
`value`	optional	If given, the variable content is checked against the `value` parameter. Only if the variable content does not match the given value, the enclosed tags are added. If this parameter is not given, the enclosed tags are only added if the variable is not defined or does not contain any text.
`match`	optional	The `match` parameter can be used instead of the `value` parameter to specify a regular expression instead of a fixed value. The enclosed tags are only added if the variable contents do not match the regular expression exactly. This parameter is only used when `value` is not given.
`matchoptions`	optional	Comma separated modification options for the regular expression evaluation. The most useful option is `IGNORECASE` which will force a case insensitive match of the pattern. This parameter is only used when `match` is given.

If you use regular expressions make sure that the regular expression is designed correctly.

See section Regular expressions for details about regular expressions.

Samples

Samples

```
<ifnot name="email">
     <exception>Email address is not defined</exception>
</ifnot>
```

```
<ifnot name="CN" match="^.+\.keyon\.ch$"
       matchoptions="IGNORECASE">
     <exception>
          CN given (<insert name="CN"/>) doesn’t end in .keyon.ch
     </exception>
</ifnot>
```

This construct would prevent the use of a CN which does not end in *.keyon.ch*.

ifcontains

Tag

Tag name	Enclosed content
`ifcontains`	Only enclosed tags are allowed. The enclosed tags will be added to the custom extension if the content of the specified element variable is present in the tokenized content of the list variable.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable to check for the presence of `element`'s content after tokenizing its own content.
`delimiters`	optional	The delimiters to use when tokenizing the content of `name`. Multiple delimiters including the space character can be specified. If not defined, the variable `name` will be tokenized at any white space character.
`element`	mandatory	The variable whose content must be present in the tokenized content of `name`.
`ignorecase`	optional	Defines if the lookup in the tokenized list is performed case insensitive (`true`) or if an exact match is required (`false`). If not defined, a case insensitive match is used (`true`).

Samples

Samples

```
<define name="invalidDomains">
      google.com
      microsoft.com
      paypal.com
</define>
<define name="domain">
      <requestattribute name="domain"/>
</define>
<ifcontains name="invalidDomains" element="domain">
      <exception>Restricted domain specified</exception>
</ifcontains>
```

This construct would ensure that the request attribute *domain* is not in the list of invalid domains, i.e. is not *google.com*, *microsoft.com* or *paypal.com*.

ifnotcontains

Tag

Tag name	Enclosed content
`ifnotcontains`	Only enclosed tags are allowed. The enclosed tags will be added to the custom extension if the content of the specified element variable is not present in the tokenized content of the list variable.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable to check for the absence of `element`'s content after tokenizing its own content.
`delimiters`	optional	The delimiters to use when tokenizing the content of the variable `name`. Multiple delimiters including the space character can be specified. If not defined, the variable `name` will be tokenized at any white space characters.
`element`	mandatory	The variable whose content must not be present in the tokenized content of `name`.
`ignorecase`	optional	Defines if the lookup in the tokenized list is performed case insensitive (`true`) or if an exact match is required (`false`). If not defined, a case insensitive match is used (`true`).

Samples

Samples

```
<define name="validDomains">
      keyon.ch
      keyoff.ch
</define>
<define name="domain">
      <requestattribute name="domain"/>
</define>
<ifnotcontains name="validDomains" element="domain">
      <exception>Domain specified not owned by keyon AG</exception>
</ifnotcontains>
```

This construct would ensure that the request attribute *domain* is contained in the list of valid domains, i.e. is either *keyon.ch* or *keyoff.ch*.

exception

Tag

Tag name	Enclosed content
`exception`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. If a text is given, the text is used as the exception message. Note that the creation of the exception is aborted when an `exception` tag is encountered. `exception` tags are therefore used within `if` or `ifnot` tags.

You can use the exception tag to prevent the issuance of a certificate if required data (e.g. specific request attribute set) is missing.

Together with the possibility of preventing empty extensions to be included, this mechanism can also be used to ensure that all required attributes or RDN elements for the subject DN building are present.

See *Using a custom extension definition to enforce subject name requirements* for an example of how to implement such an extension.

Parameters

Tag parameter	Presence	Description
–	-	-

Samples

Samples

```
<ifnot name="email">
      <exception>Email address is not defined</exception>
</ifnot>
```

Loop Tags

foreach

Tag

Tag name	Enclosed content
`foreach`	Only enclosed tags are allowed. The content of the specified variable is tokenized at the given delimiters and for each token found, a temporary variable with the token content is defined and the enclosed tags are evaluated.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The variable content to tokenize. If the variable is not defined, the enclosed tags are not evaluated.
`delimiters`	mandatory	The delimiters to use when tokenizing the variable content. Multiple delimiters including the space character can be specified.
`define`	mandatory	The name of the temporary variable to define with the token content. Heading and trailing whitespace of the token content is removed. This variable is only available to the enclosed tags.
`reverse`	optional	If `true`, reverses the order in which the tokens found after tokenization are passed to the enclosed tags. If not defined or `false`, tokens are processed in the order they appear after tokenization.

Make sure that the name of the temporary variable specified with `define` is unique and does not overwrite any other existing variable.

Samples

Samples

```
<define name="dnsNamesVar">
    <requestattribute name="dnsNames"/>
</define>
<sequence includeempty="false">
    <foreach name="dnsNamesVar" delimiters=";" define="myDnsName">
        <ia5string tag="2">
            <insert name="myDnsName"/>
        </ia5string>
    </foreach>
</sequence>
```

This construct would create a sequence of *dNSName* entries for a SubjectAltName extension by tokenizing the ; delimited contents of the *dnsNames* request attribute.

foreachrequestrdn

Tag

Tag name	Enclosed content
`foreachrequestrdn`	Only enclosed tags are allowed. For each RDN matching the given OID in the request DN, a temporary variable with the RDN content is defined and the enclosed tags are evaluated.

Parameters

Tag parameter	Presence	Description
`oid`	mandatory	The OID identifying the RDN elements to enumerate. If no RDN with the given OID is present, the enclosed tags are not evaluated. See `requestrdn` for a list of OIDs for the most common RDN elements.
`define`	mandatory	The name of the temporary variable to define with the RDN content. This variable is only available to the enclosed tags.
`reverse`	optional	If `true`, reverses the order in which the RDNs are found in the request DN are passed to the enclosed tags. If not defined or `false`, RDNs are processed in the order they appear in the request DN.

Make sure that the name of the temporary variable specified with `define` is unique and does not overwrite any other existing variable.

Samples

Samples

```
<foreachrequestrdn oid="2.5.4.3" define="myCN">
    <ifnot name="myCN" match="^.+\.keyon\.test$"
           matchoptions="IGNORECASE">
        <exception>
            CommonName '<insert name="myCN"/>' doesn't
            end in .keyon.test
        </exception>
    </ifnot>
</foreachrequestrdn>
```

This construct would ensure that all *CommonName* entries in the request DN contain a fully qualified DN ending in *.keyon.test*.

foreachsubjectrdn

Tag

Tag name	Enclosed content
`foreachsubjectrdn`	Only enclosed tags are allowed. For each RDN matching the given OID in the subject DN, a temporary variable with the RDN content is defined and the enclosed tags are evaluated. The subject DN may be different from the request DN if the subject DN building of true-Xtender is used. The subject DN is the DN that will be included in the certificate and may differ from the request DN.

Parameters

Tag parameter	Presence	Description
`oid`	mandatory	The OID identifying the RDN elements to enumerate. If no RDN with the given OID is present, the enclosed tags are not evaluated. See `requestrdn` for a list of OIDs for the most common RDN elements.
`define`	mandatory	The name of the temporary variable to define with the RDN content. This variable is only available to the enclosed tags.
`reverse`	optional	If `true`, reverses the order in which the RDNs are found in the request DN are passed to the enclosed tags. If not defined or `false`, RDNs are processed in the order they appear in the request DN.

Make sure that the name of the temporary variable specified with `define` is unique and does not overwrite any other existing variable.

Samples

Samples

```
<foreachsubjectdn oid="2.5.4.3" define="myCN">
    <ifnot name="myCN" match="^.+\.keyon\.test$"
           matchoptions="IGNORECASE">
        <exception>
            CommonName '<insert name="myCN"/>' doesn't
            end in .keyon.test
        </exception>
    </ifnot>
</foreachsubjectrdn>
```

This construct would ensure that all *CommonName* entries in the request DN contain a fully qualified domain name ending in *.keyon.test*.

foreachsubjectaltname

Tag

Tag name	Enclosed content
`foreachsubjectaltname`	Only enclosed tags are allowed. For each element of the given type in the *SubjectAltName*, a temporary variable with the entry content is defined and the enclosed tags are evaluated.

Parameters

Tag parameter	Presence	Description
`type`	mandatory	The type of *SubjectAltName* element to enumerate. If no element with the given OID is present, the enclosed tags are not evaluated. See `subjectaltname` for a list of available types.
`define`	mandatory	The name of the temporary variable to define with the *SubjectAltName* elements of the given type. This variable is only available to the enclosed tags.
`reverse`	optional	If `true`, reverses the order in which the elements of the given type found in the *SubjectAltName* are passed to the enclosed tags. If not defined or `false`, elements are processed in the order they appear in the *SubjectAltName*.

Make sure that the name of the temporary variable specified with `define` is unique and does not overwrite any other existing variable.

Samples

Samples

```
<foreachsubjectaltname type="dNSName" define="myDnsName">
    <ifnot name="myDnsName" match="^.+\.keyon\.test$"
           matchoptions="IGNORECASE">
        <exception>
            SubjectAltName policy violation: dNSName
            '<insert name="myDnsName"/>' doesn't end
            in .keyon.test
        </exception>
    </ifnot>
</foreachsubjectaltname>
```

This construct ensures that all *dNSName* entries present in the *SubjectAltName* are fully qualified domain names ending in *.keyon.test*.

```
<foreachsubjectaltname type="ipAddress" define="myIpAddress">
    <exception>
        SubjectAltName may not contain ipAddress entries
    </exception>
</foreachsubjectaltname>
```

This construct ensures that no *ipAddress* is present in the *SubjectAltName*.

```
<foreachsubjectaltname type="otherNameOID" define="myOID">
    <ifnot name="myOID" value="1.3.6.1.4.1.311.20.2.3">
        <exception>
            SubjectAltName may not contain otherName entries
            other than UPN
        </exception>
    </ifnot>
</foreachsubjectaltname>
```

This construct ensures that the only *otherName* entries present in the *SubjectAltName* are of type *user principal name*.

Request Data Tags

requestattribute

Tag

Tag name	Enclosed content
`requestattribute`	Text or tags that insert text are ignored. No enclosed ASN.1 tags are allowed. The content of the named request attribute is inserted. Note that an empty string is inserted if the request attribute is not set.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The request attribute whose content to insert.
`ignorecase`	optional	Defines if the request attribute name is matched case insensitive (`true`) or if an exact match is required (`false`). If not defined, the case must match exactly (`false`).

Samples

Samples
`<ia5string> <requestattribute name="email"/> </ia5string>`

Request attributes are attributes that are passed along with request but are not part of the signed request itself.

definerequestattribute

Tag

Tag name	Enclosed content
`definerequestattribute`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be assigned to the request attribute but only in the scope of the true-Xtender request processing, i.e. the attribute set is not stored in the certificate services database. You can however access such a defined attribute for the true-Xtender DN building.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The request attribute to define or overwrite.

Samples

Samples

```
<definerequestattribute name="newCommonName">
   <subjectaltname type="dnsName"/>
</definerequestattribute>
```

Request attributes defined using `definerequestattribute` are only available in true-Xtender custom extensions and for subject DN building with true-Xtender.

Note that the request attribute name specified in DN building using `<RequestAttribute(`*name*`)>` is case sensitive. Make sure that the name specified in the `name` attribute of `definerequestattribute` matches the name given in `requestattribute` exactly.

requestproperty

Tag

Tag name	Enclosed content
`requestproperty`	Text or tags that insert text are ignored. No enclosed ASN.1 tags are allowed. The content of the named request property is inserted. Note that an empty string is inserted if the request property is not set.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The request property whose content to insert.

The following table lists the request properties that are available using the `requestproperty` tag. Note however that not all request properties may be available for every request:

Request Property Name	Description
`Request.CallerName`	The user or machine context that submitted the certificate request to the CA.
`Request.CommonName`	The common name attribute of the DN from the Subject of the certificate request.
`Request.Country`	The country attribute of the DN from the Subject of the certificate request.
`Request.DeviceSerialNumber`	The device serial number attribute of the DN from the Subject of the certificate request.
`Request.Disposition`	The request disposition code.
`Request.Disposition.Message`	The text description of Request.Disposition. Request.Disposition.Message is for presentation to a user and can contain any text string, including NULL, that the implementer considers informative.
`Request.DistinguishedName`	The distinguished name (DN) from the Subject attribute of the certificate request.

Request Property Name	Description
`Request.DomainComponent`	The domainComponent attribute of the DN from the Subject of the certificate request.
`Request.EMail`	The EmailAddress attribute of the DN from the Subject of the certificate request.
`Request.GivenName`	The given name (also called first name) attribute of the DN from the Subject of the certificate request.
`Request.Initials`	The initials attribute of the DN from the Subject of the certificate request.
`Request.Locality`	The locality attribute of the DN from the Subject of the certificate request.
`Request.Officer`	Indicates whether the caller is the certificate manager of the entity that corresponds to the Request.RequesterName.
`Request.OrgUnit`	The organizational-unit attribute of the DN from the Subject of the certificate request.
`Request.Organization`	The organization attribute of the DN from the Subject of the certificate request.
`Request.RequestAttributes`	The certificate request attributes as defined in [MS-WCCE].
`Request.RequestFlags`	Additional certificate request information.
`Request.RequestID`	The request ID.
`Request.RequestType`	The type or format of a certificate request, such as PKCS#10 or the Cryptographic Message Syntax (CMS) standard with Common Messaging Calls (CMC) as specified in [RFC2797].
`Request.RequesterName`	The RequesterName that is included in the certificate request.
`Request.SignerApplicationPolicies`	The list of valid Extended Key Usage OIDs for each signer certificate from the certificate request.

Request Property Name	Description
`Request.SignerPolicies`	The list of valid certificate policy OIDs for each signer certificate from the certificate request.
`Request.State`	The state or province name attribute of the DN from the Subject of the certificate request.
`Request.StreetAddress`	The street address attribute of the DN from the Subject of the certificate request.
`Request.SurName`	The surname attribute of the DN from the Subject of the certificate request.
`Request.Title`	The title attribute of the DN from the Subject of the certificate request.
`Request.UnstructuredAddress`	The unstructured address attribute of the DN from the Subject of the certificate request.
`Request.UnstructuredName`	The unstructured name attribute of the DN from the Subject of the certificate request.

Depending on how the request was created, most of the properties will not be set. The most useful properties is `Request.RequesterName` which contains the user raising the request in the form `DOMAIN\userid`.

In addition to the standard request properties available to a policy module, the keyon true-Xtender extracts information from the certificate signing request (CSR) and makes them available as request properties.

The type of the CSR is available in the following property:

Request Property Name	Description
`Request.CSR.Type`	The type of the certificate signing request. The following types are defined: PKCS7, PKCS10, CMC or KEYGEN.

Note that the properties in the CSR are added by the client as part of the request and may not contain authenticated information. Using specific software, a client could easily forge e.g. the REQUEST_CLIENT_INFO to include arbitrary user and machine names.

If the request has an OS_VERSION attribute (OID 1.3.6.1.4.1.311.13.2.3) present that contains version information about the client operating system on which the certificate request was generated, the contents is available as a string in the following property:

Request Property Name	Description
`Request.CSR.Attribute.OSVersion`	The client operating system version information..

If the request has a REQUEST_CLIENT_INFO attribute (OID 1.3.6.1.4.1.311.21.20) present that can be used to identify the client that generated a certificate request, the contents are available in these properties:

Request Property Name	Description
`Request.CSR.Attribute.ClientInfo.ClientId`	The type of client application that generated the request.
`Request.CSR.Attribute.ClientInfo.MachineName`	The Domain Name System (DNS) name of the computer that generated the request.
`Request.CSR.Attribute.ClientInfo.ProcessName`	The name of the application that generated the request.
`Request.CSR.Attribute.ClientInfo.UserName`	The Security Accounts Manager (SAM) name of the user.

If the request has an ENROLLMENT_CSP_PROVIDER attribute (OID 1.3.6.1.4.1.311.13.2.2) present that identifies the cryptographic provider used by the entity requesting the certificate, the contents are available in these properties:

Request Property Name	Description
`Request.CSR.Attribute.EnrollmentCSP.CSPName`	The provider name.
`Request.CSR.Attribute.EnrollmentCSP.KeySpec`	A value that identifies whether the key pair stored by the provider or key container is used for encryption or for signing content.

If the request has an ENROLLMENT_NAME_VALUE_PAIR attribute (OID 1.3.6.1.4.1.311.13.2.1) present that contains generic name value pairs, the name value pairs are available in these properties:

Request Property Name	Description
`Request.CSR.Attribute.NameValuePair.`*`name`*	Where *`name`* is the name of the value.

One of the well-known name value pairs is the certificate template requested by the client which is available in the property `Request.CSR.Attribute.NameValuePair.CertificateTemplate`.

If the request is of type KEYGEN or has a Challenge Password according to PKCS#9 (OID 1.2.840.113549.1.9.7) present, the contents is available as a string in the following property:

Request Property Name	Description
`Request.CSR.Attribute.ChallengePassword`	The challenge password.

If the request was processed by the Network Device Enrollment Service (NDES), i.e. the request was posted using the SCEP protocol, the `Request.CSR.Attribute.ChallengePassword` contains the password created by NDES.

Together with the `adsquery` tag and the subject name building capabilities of true-Xtender, this property can be used to ensure that a network device cannot get a certificate for a common name other than the one intended for this specific device. It is thus possible with little additional effort to mitigate the known vulnerability in SCEP documented by the US-CERT Vulnerability Note VU#971035: Simple Certificate Enrollment Protocol (SCEP) does not strongly authenticate certificate requests

Other request attributes that consist of only string content are available as a string in the following property:

Request Property Name	Description
`Request.CSR.Attribute.`*`oid`*	Where *oid* is the OID of the string attribute.

Samples

Samples

```
<ia5string>
    <requestproperty name="Request.RequesterName"/>
</ia5string>
```

```
<define name="CSRType">
    <requestproperty name="Request.CSR.Type"/>
</define>
<ifnot name="CSRType" value="PKCS10">
     <exception>Only PKCS#10 request types allowed</exception>
</ifnot>
```

certificateproperty

Tag

Tag name	Enclosed content
`certificateproperty`	Text or tags that insert text are ignored. No enclosed ASN.1 tags are allowed. The content of the named certificate property is inserted. Note that an empty string is inserted if the certificate property is not set.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The certificate property whose content to insert.

The following table lists the certificate properties that are available using the `certificateproperty` tag. Note however that not all certificate properties may be available for every request:

Certificate Property Name
`CertificateTemplate`
`EnrollmentFlags`
`GeneralFlags`
`PublicKeyAlgorithm`
`PublicKeyLength`
`UPN`

You can use the `PublicKeyLength` variable and an `ifnot` tag to ensure a certain public key size.

Samples

Samples

```
<ia5string>
    <certificateproperty name="CertificateTemplate"/>
</ia5string>
```

requestrdn

Tag

Tag name	Enclosed content
`requestrdn`	Text or tags that insert text are ignored. No enclosed ASN.1 tags are allowed. The content of the specified request RDN is inserted.

Parameters

Tag parameter	Presence	Description
`oid`	mandatory	The OID of the RDN who's content to insert.
`element`	optional	If more than one RDN with the given OID exists, a specific RDN can be selected. If not defined, the first (1) RDN is inserted.

OIDs for common RDN elements

Long name	Short name	OID
CommonName	CN	`2.5.4.3`
Country	C	`2.5.4.6`
DeviceSerialNumber	-	`2.5.4.5`
DomainComponent	DC	`0.9.2342.19200300.100.1.25`
Email	E	`1.2.840.113549.1.9.1`
GivenName	G	`2.5.4.42`
Initials	I	`2.5.4.43`
Locality	L	`2.5.4.7`
Organization	O	`2.5.4.10`
OrganizationalUnit	OU	`2.5.4.11`
State	ST, S	`2.5.4.8`
StreetAddress	Street	`2.5.4.9`
SurName	SN	`2.5.4.4`

Long name	Short name	OID
Title	T	2.5.4.12

Samples

Samples
<ia5string> <requestrdn oid="2.5.4.3" element="1"/> </ia5string>

subjectrdn

Tag

Tag name	Enclosed content
`subjectrdn`	Text or tags that insert text are ignored. No enclosed ASN.1 tags are allowed. The content of the specified subject RDN is inserted. Note that an empty string is inserted if the subject RDN is not set. The subject RDN may be different from the request RDN if the subject DN building of true-Xtender is used. The subject DN is the DN that will be included in the certificate and may differ from the request DN.

Parameters

Tag parameter	Presence	Description
`oid`	mandatory	The OID of the RDN who's content to insert.
`element`	optional	If more than one RDN with the given OID exists, a specific RDN can be selected. If not defined, the first (1) RDN is inserted.

See `requestrdn` for a list of OIDs for the most common RDN elements.

Samples

Samples
`<ia5string> <subjectrdn oid="2.5.4.3" element="1"/> </ia5string>`

subjectaltname

Tag

Tag name	Enclosed content
`subjectaltname`	Text or tags that insert text are ignored. No enclosed ASN.1 tags are allowed. The content of the specified *SubjectAltName* element is inserted.

Parameters

Tag parameter	Presence	Description
`type`	mandatory	The type of the *SubjectAltName* element who's content to insert. See the table below for a list of available types.
`element`	optional	If more than one element of the given type exists, a specific element can be selected. If not defined, the first (1) element is inserted.

SubjectAltName Element Types

Type	Tag	Format	Description
`otherNameOID`	–	String	This is an artificial type which will return all OIDs of present *otherName* entries in the *SubjectAltName*. (Useful in conjunction with the `foreachsubjectaltname` tag.)
`otherName[<oid>]`	0	String or Hex String	*otherName* entry matching the given OID in brackets either as a string in case the value represents a string or a hex string representing the raw ASN.1 DER encoding if the value is not a string.
`rfc822Name`	1	String	*rfc822Name* entry as a string.
`dnsName`	2	String	*dNSName* entry as a string.
`directoryName`	4	Hex String	*directoryName* as a hex string representing the ASN.1 DER encoded *Name*.
`url`	6	String	*uniformResourceIdentifier* entry as a string.
`ipAddress`	7	String	*iPAddress* entry as a string.
`registeredID`	8	OID String	*registeredID* entry as an OID string.

Samples

Samples

```
<utf8string>
    <subjectaltname type="otherName[1.3.6.1.4.1.311.20.2.3]"/>
</utf8string>
```

timenow

Tag

Tag name	Enclosed content
`timenow`	Text or tags that insert text which form a date format. No enclosed ASN.1 tags are allowed. The current time is formatted according to the enclosed format. If the tag is enclosed in an ASN.1 `utctime` or `generalizedtime` tag, the correct date format string for the ASN.1 date is inserted.

Format string elements

Placeholder	Description
`%a`	Abbreviated weekday name
`%A`	Full weekday name
`%b`	Abbreviated month name
`%B`	Full month name
`%c`	Date and time representation appropriate for locale
`%d`	Day of month as decimal number (01 – 31)
`%H`	Hour in 24-hour format (00 – 23)
`%I`	Hour in 12-hour format (01 – 12)
`%j`	Day of year as decimal number (001 – 366)
`%m`	Month as decimal number (01 – 12)
`%M`	Minute as decimal number (00 – 59)
`%p`	Current locale's A.M./P.M. indicator for 12-hour clock
`%S`	Second as decimal number (00 – 59)
`%U`	Week of year as decimal number, with Sunday as first day of week (00 – 53)
`%w`	Weekday as decimal number (0 – 6; Sunday is 0)
`%W`	Week of year as decimal number, with Monday as first day of week (00 – 53)

Placeholder	Description
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z, %Z	Time-zone name or abbreviation; no characters if time zone is unknown

E.g. the format string `%d.%m.%Y %H:%M:%S` will create a date string like `18.01.2007 23:00:00`. For the ASN.1 `GeneralizedTime` tag, the format to use would be `%Y%m%d%H%M%SZ`. For the ASN.1 `UTCTime` tag, the format to use would be `%y%m%d%H%M%SZ`.

Parameters

Tag parameter	Presence	Description
`offset`	optional	A positive or negative offset in seconds that is added to the current time before formatting the date.
`round`	optional	If defined, the time in with optional added offset is rounded to the next multiple of `round`. If `round` is e.g. 60, the seconds part of the inserted date will always be 0. If offset is set to -3600 and round set to 3600, the date string will contain the latest full hour before the current date.

Samples

Samples
`<generalizedtime> <timenow/> </generalizedtime>`

`<ia5string>`
`  <timenow offset="31536000">%d.%m.%Y %H:%M:%S</timenow>`
`</ia5string>`

timenowutc

Tag

Tag name	Enclosed content
`timenowutc`	Text or tags that insert text which form a date format. No enclosed ASN.1 tags are allowed. The current time in UTC is formatted according to the enclosed format. If the tag is enclosed in an ASN.1 `utctime` or `generalizedtime` tag, the correct date format string for the ASN.1 date is inserted.

See `timenow` for format string elements. E.g. the format string `%d.%m.%Y %H:%M:%S` will create a date string like `18.01.2007 23:00:00`

For the ASN.1 `GeneralizedTime` tag, the format to use would be `%Y%m%d%H%M%SZ`. For the ASN.1 `UTCTime` tag, the format to use would be `%y%m%d%H%M%SZ`.

Parameters

Tag parameter	Presence	Description
`offset`	optional	A positive or negative offset in seconds that is added to the current time before formatting the date.
`round`	optional	If defined, the time in UTC with optional added offset is rounded to the next multiple of `round`. If `round` is e.g. 60, the seconds part of the inserted date will always be 0. If offset is set to -3600 and round set to 3600, the date string will contain the latest full hour before the current date.

Samples

Samples
`<generalizedtime> <timenowutc/> </generalizedtime>`
`<timenowutc offset="31536000">%d.%m.%Y %H:%M:%S</timenowutc>`

Data Lookup Tags

adsquery

Tag

Tag name	Enclosed content
`adsquery`	Only `placeholder` tags are permitted. No enclosed ASN.1 tags are allowed. Query an Active Directory Service using a search filter and insert the contents of a specified attribute if a matching entry is found.

Parameters

Tag parameter	Presence	Description
`binding`	optional	The server / domain and/or base path to use for the query using LDAP ADsPath syntax. Defaults to: `LDAP://`*domain* where *domain* is the Active Directory Domain where the Certificates Services Server is a member of.
`username`	optional	The user name to use for authentication against the Active Directory. If not specified, the credentials of the certificates services process are used for authentication. The `username` is only required if the Certificates Services server is not part of the Active Directory domain to query.
`password`	optional	The password to use for authentication against the Active Directory. See `username` for details.
`secureauthentication`	optional	If an authentication with `username` / `password` is used, specifies if only a secure authentication mechanism is used (`true`, default) or not (`false`). AD LDS and other LDAP servers may require this option to be set to `false` for authentication to work.

Tag parameter	Presence	Description
`scope`	optional	The search scope. May be `base`, `onelevel` or `subtree` (default). If you do not specify `subtree` as the search scope you must specify an object path in the `binding` attribute.
`resultrow`	optional	The index of the result row to use. The first row uses index 1. Defaults to `1`.
`resultmaxcount`	optional	The maximum result rows the query may return to succeed. If only a single row is expected (normal case), 1 must be specified so that an exception is thrown if the search returns more than one matching entry. Defaults to `1`.
`attributerow`	optional	The index of the attribute value in the matching entry to use in case of multi valued attributes. The first attribute value uses index 1. Defaults to `1`.
`attributeorder`	optional	The sorting order of the attribute values. Can be `asc` for ascending order (default) or `desc` for descending order. Used to ensure that order of attributes returned is always the same if multiple queries are executed to get all attribute values.
`attributemaxcount`	optional	The maximum number of values the specified attribute in the matching entry may contain. If the attribute is a single value attribute (normal case), 1 must be specified so that an exception is thrown if the attribute contains more than one value. Defaults to `1`.
`attribute`	mandatory	The attribute name whose value to insert upon success. You can specify ordinary (e.g. `userPrincipalName`) as well as constructed (e.g. `distinguishedName`) attributes. Note that only attributes that are of string form can be inserted.

Tag parameter	Presence	Description
`searchfilter`	mandatory	The LDAP search filter to use for the query. The filter uses LDAP filter syntax and can contain placeholders of the form `{name}` that are quoted and inserted into the filter before the query is executed. Note that you must insert the `&` operator as `&` due to XML restrictions. Some servers may not work correctly if the filter string is split over multiple lines.

An exception will be thrown if the Active Directory Services is not available, the binding information is incorrect or the search filter is invalid.

If not results are found matching the filter, not exception is thrown and an empty string is inserted into the enclosing tag. You can use the `define` and `ifnot` tags to detect this condition.

To define the contents for {placeholders} in the search filter, you can either define variables before using `adsquery`, or add `placeholder` tags to the `adsquery` tag. If a `placeholder` tag is used, a local variable is defined that is only visible in the scope of the `adsquery` tag.

Sub-Tag

Tag name	Enclosed content
`placeholder`	Only string contents and tags that insert string contents are allowed. No enclosed ASN.1 tags are allowed. Insert the contents into the placeholder in the search filter. Whitespace at the beginning and end of the contents is removed.

Parameters

Tag parameter	Presence	Description
`name`	mandatory	The name of the placeholder in the search filter which will be replaced by the contents.

If the contents of a `placeholder` tag evaluate to an empty string, a variable with the same name is inserted instead if defined. If a placeholder in the search filter cannot be replaced with a non-empty string, an exception will be thrown.

Binding strings

The LDAP ADsPath (see [ADS-1]) requires the following format for the binding string:

```
LDAP://HostName[:PortNumber][/DistinguishedName]
GC://HostName[:PortNumber][/DistinguishedName]
```

Note: The left and right bracket characters ([ ]) indicate optional parameters; it is not a literal part of the binding string. The protocol (LDAP / GC) string is case sensitive.

- The `HostName` can be a computer name, an IP address, or a domain name. A server name can also be specified in the binding string.
- The `PortNumber` specifies the port to be used for the connection. The default port number is 389 if not using an SSL connection or 636 if using an SSL connection.
- The `DistinguishedName` specifies the distinguished name of a specific object. A distinguished name for a given object is guaranteed to be unique.

The following table gives some examples of binding strings.

LDAP ADsPath example	Description
`GC://keyon.ch`	Bind to a Global Catalog server in the domain keyon.ch.
`LDAP://keyon.ch`	Bind to an Active Directory server in the domain keyon.ch.
`LDAP://server01`	Bind to a specific server.
`LDAP://server01:390`	Bind to a specific server using the specified port number.
`LDAP://CN=users,DC=keyon,DC=ch`	Bind to a specific object.
`LDAP://server01/CN=users,DC=keyon,DC=ch`	Bind to a specific object through a specific server.

Not all attributes are available in the Global Catalog. Please check the Microsoft Software Developers Network (MSDN) for details which attributes are available in the Global Catalog and which are only available using Active Directories Services.

Search filters

Please consult [RFC2254] and [ADS-2] for details on the LDAP search filter syntax. The string representation of an LDAP search filter is defined by the following grammar. The filter format uses a prefix notation:

```
filter       = "(" filtercomp ")"
filtercomp   = and / or / not / item
and          = "&" filterlist
or           = "|" filterlist
not          = "!" filter
filterlist   = 1*filter
item         = simple / present / substring / extensible
simple       = attr filtertype value
filtertype   = equal / approx / greater / less
equal        = "="
approx       = "~="
greater      = ">="
less         = "<="
extensible   = attr [":dn"] [":" matchingrule] ":=" value
               / [":dn"] ":" matchingrule ":=" value
present      = attr "=*"
substring    = attr "=" [initial] any [final]
initial      = value
any          = "*" *(value "*")
final        = value
attr         = AttributeDescription
matchingrule = MatchingRuleId
value        = AttributeValue
```

A substring `{x}` in the search filter where *x* is a text string is replaced by the contents of the variable or placeholder *x*.

Variables inserted into the search filter are quoted appropriately. The `&` operator in a search filter string however must be inserted as `&` due to XML requirements.

You can use the `memberOf` attribute in a search filter to check if a user is member of specific group:

```
(memberOf=CN=Administrators,CN=Builtin,DC=keyon,DC=ch)
```

If the user is not a member of the `Administrators` group, no value for an attribute (e.g. `distinguishedName`) is returned. You can use `define` and `ifnot` tags to raise an exception in this case and abort the issuance of the certificate with an exception.

Testing your search filters using dsquery

You can use the `dsquery` command line tool that is part of the Windows Server installation to engineer and check your search filters.

Sample (all on one line):

```
> dsquery * domainroot -attr userPrincipalName
    -filter "(&(sAMAccountName=user1)(objectCategory=person)(objectClass=user))"
```

Output if the user is found and the *userPrincipalName* attribute is defined:

```
userPrincipalName
user1@keyon.ch
```

Don't forget to enter a `&` operator in a search filter as `&` in the true-Xtender XML file since the ampersand is a reserved character in XML.

Samples

Samples

```
<adsquery
    attribute="userPrincipalName"
    searchfilter="(&amp;(sAMAccountName={sAMAccountName})
                  (objectCategory=person)(objectClass=user))"/>
```

This will search a user record in the domain where the Certificate Services server is a member of with a *sAMAccountName* attribute value that is equal to the contents of the variable *sAMAccountName*. If exactly one user record is found and the *userPrincipalName* attribute contains a single value, the contents of the *userPrincipalName* are inserted to the enclosing tag.

```
<adsquery
    attribute="distingishedName"
    searchfilter="(&amp;(sAMAccountName={1})
                  (objectCategory=person)(objectClass=user))">
    <placeholder name="1">
        <requestattribute name="logonName"/>
    </placeholder>
</adsquery>
```

This will search a user record in the domain where the Certificate Services server is a member of with a *sAMAccountName* attribute value that is equal to the contents of the request attribute *logonName*. If exactly one user record is found the contents of the constructed *distinguishedName* attribute are inserted to the enclosing tag.

Text Transformation Tags

toupper

Tag

Tag name	Enclosed content
`toupper`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be converted to uppercase.

Parameters

Tag parameter	Presence	Description
–	-	-

Samples

Samples
`<toupper> <requestattribute name="logonid"/> </toupper>`

tolower

Tag

Tag name	Enclosed content
`tolower`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be converted to lowercase.

Parameters

Tag parameter	Presence	Description
–	-	-

Samples

Samples
`<tolower> <requestattribute name="email"/> </tolower>`

reversedn

Tag

Tag name	Enclosed content
`reversdn`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be considered a distinguished name and the order of the relative distinguished elements is reversed.

Parameters

Tag parameter	Presence	Description
–	-	-

If the expanded text is not a distinguished name, depending on the structure either an empty string is inserted or an exception is thrown.

Samples

Samples
`<reversedn> CN=User, DC=keyon, DC=ch </reversedn>` Would insert the text `DC=ch, DC=keyon, CN=User` as the result of the reverse DN operation.

iptohexstring

Tag

Tag name	Enclosed content
`iptohexstring`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be considered an IPv4 or IPv6 address and the binary representation of the IP address in network byte order is returned as a hex string e.g. for use in the `octetstring` tag. The type of IP address (v4 or v6) is detected automatically and the generated hex string represents either 4 bytes (IPv4) or 16 bytes (IPv6).

Parameters

Tag parameter	Presence	Description
–	-	-

An exception is thrown if the expanded text is not a valid IPv4 or IPv6 address. Note that for IPv4 addresses, the numbers are treated as octal (base 8) if leading zeroes are used. `010.020.030.040` is thus not the same as `10.20.30.40`:

`010.020.030.040` ⇨ `08101820`

`10.20.30.40` ⇨ `0A141E28`

Samples

Samples

```
<octetstring tag="7">
     <iptohexstring><requestattribute name="ip"/></iptohexstring>
</octetstring>
```

Would insert a tagged `octetstring` with the binary representation of the IP address stored in the request attribute `ip`.

`10.20.30.40` ⇨ `0A141E28`

`FE80::0202:b3ff:fe1e:8329` ⇨ `FE80000000000000202B3FFFE1E8329`

uuidtohexstring

Tag

Tag name	Enclosed content
`uuidtohexstring`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be considered a UUID and the binary representation of the UUID is returned as a hex string e.g. for use in the `octetstring` tag.

Parameters

Tag parameter	Presence	Description
–	-	-

An exception is thrown if the expanded text is not a valid UUID of the form

`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

Note that the UUID may optionally be enclosed in {}.

Samples

Samples

```
<explicit tag="0">
      <objectid>1.3.6.1.4.1.311.25.1</objectid>
      <explicit tag="0">
            <octetstring>
                  <uuidtohexstring>
                        <requestattribute name="guid"/>
                  </uuidtohexstring>
            </octetstring>
      </explicit>
</explicit>
```

Would insert an *otherName* entry for a Domain Controller with the GUID stored in the request attribute `guid`.

```
6F9EF915-B89A-48FB-A696-0E12E80588F4 ⇨
15F99E6F9AB8FB48A6960E12E80588F4
```

base64tohexstring

Tag

Tag name	Enclosed content
`base64tohexstring`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text will be considered Base64 encoded binary data and the decoded binary data is returned as a hex string e.g. for use in the `octetstring` or `raw` tag.

Parameters

Tag parameter	Presence	Description
–	-	-

An exception is thrown if the expanded text is not valid Base64 encoded data.

Samples

Samples

```
<explicit tag="0">
      <raw>
            <base64tohexstring>
                  <requestattribute name="asn1element"/>
            </base64tohexstring>
      </raw>
</explicit>
```

Would insert an explicit tagged ASN.1 element which is provided in Base64 encoded form in the request attribute `asn1element`.

ASN.1 Basic Types

objectid

Tag

Tag name	Enclosed content
`objectid`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text must be a valid OID.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<objectid> 1.2.840.113549.1.1.5 </objectid>`
Generated ASN.1 representation:
`OBJECT IDENTIFIER '1 2 840 113549 1 1 5'`

`<objectid tag="5"> 20 </objectid>`
Generated ASN.1 representation:
`[5] 1.2.840.113549.1.1.5`
dumpasn1 will show such an encoding as
`[5]`
`2A 86 48 86 F7 0D 01 01 05`

`<objectid> <requestattribute name="algorithm"/> </objectid>`

character string types

Tag

Tag name	Enclosed content

`ia5string`
`numericstring`
`universalstring`
`bmpstring`
`utf8string`
`teletexstring`
`visiblestring`
`generalstring`
`printablestring`
	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. Note that depending on the chosen string type, character set limitations apply. Consult an ASN.1 reference manual for details.
	Leading and trailing whitespace is removed. You may need to use the XML syntax `<![CDATA[ text ]]>` If you want to enter static special characters.
	Note that the more esoteric character string types like `VideotexString` are not supported as they are not used with X.509 extensions.

You must ensure that the character string given uses only valid characters for the selected character string type.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<printablestring> 123456789abcdef </printablestring>`
Generated ASN.1 representation:
`PrintableString 'keyon'`

`<ia5string tag="2">user1</ia5string>`
Generated ASN.1 representation:
`[2] 'user1'`

`<ia5string> <requestattribute name="upn"/> </ia5string>`

date string types

Tag

Tag name	Enclosed content
`utctime` `generalizedtime`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. Leading and trailing whitespace is removed. Consult the ASN.1 reference manual for details on valid date strings. Note that the special tags `timenow` and `timenowutc` can be used to insert a correctly formatted date string with the current system time.

You must ensure that the date string given represents a valid ASN.1 date string for the selected type. If you enclose a `timenow` or `timenowutc` tag, a correct date string will be used.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<generalizedtime> 20010101010203Z </generalizedtime>`
Generated ASN.1 representation:
`GeneralizedTime '20010101010203Z'`

`<utctime> <timenowutc/> </utctime>`
Inserts the current system time in UTC. E.g. for January 18th 2007, 18:26:36 local time in Switzerland (GMT+1), the following date would be inserted:
`UTCTime '070118172636Z'`

bitstring

Tag

Tag name	Enclosed content
`bitstring`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text must only contain 0 and 1 characters. The order is least significant bit first.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<bitstring> 001 </bitstring>`
Generated ASN.1 representation:
`BIT STRING 5 unused bits`
`'100'B (bit 2)`

`<bitstring> tag="5"> 001 </bitstring>`
Generated ASN.1 representation:
`[1] '100'B`
dumpasn1 will show such an encoding as
`[1]`
`05 20`

`<bitstring> <requestattribute name="keyusage"/> </bitstring>`

octetstring

Tag

Tag name	Enclosed content
`octetstring`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text must only contain `0..9` and `A..F` characters representing the binary data in hex format. Bytes may be separated with whitespace or the `:` character. If the total number of characters is odd, a `0` character is added for padding.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<octetstring> 123456789abcdef </octetstring>`
Generated ASN.1 representation:
`OCTET STRING`
`12 34 56 78 9A BC DE F0`

`<octetstring> 12:34:56:78:9a:bc:de:f0 </octetstring>`
Generated ASN.1 representation:
`OCTET STRING`
`12 34 56 78 9A BC DE F0`

`<octetstring> <requestattribute name="data"/> </octetstring>`

boolean

Tag

Tag name	Enclosed content
`boolean`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text is evaluated as follows: If the text is *true* (case insensitive) after removing heading and trailing white space, the ASN.1 boolean is set true. Any other content will set the ASN.1 boolean to false.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<boolean> true </boolean>`
Generated ASN.1 representation:
`BOOLEAN TRUE`

`<boolean tag="1"> false </boolean>`
Generated ASN.1 representation:
`[1] FALSE`
dumpasn1 will show such an encoding as
`[1]`
`00`

`<boolean> <requestattribute name="caBool"/> </boolean>`

integer

Tag

Tag name	Enclosed content
`integer`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text must be a positive integer in the range `0..2147483647`. The implementation does not support negative integers as they have no relevance for X.509 extensions.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<integer> 12 </integer>`
Generated ASN.1 representation:
`INTEGER 12`

`<integer tag="1"> 128 </integer>`
Generated ASN.1 representation:
`[1] 128`
dumpasn1 will show such an encoding as
`[1]`
`00 80`

`<integer> <requestattribute name="numberOfLicenses"/> </integer>`

enumerated

Tag

Tag name	Enclosed content
`enumerated`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text must be a positive integer in the range `0..2147483647`.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<enumerated> 10 </enumerated>`
Generated ASN.1 representation:
`ENUMERATED 10`

`<enumerated tag="1"> 20 </enumerated>`
Generated ASN.1 representation:
`[1] 20`
dumpasn1 will show such an encoding as
`[20]`
`14`

`<enumerated> <requestattribute name="option"/> </enumerated>`

null

Tag

Tag name	Enclosed content
`null`	Ignored.

Parameters

Tag parameter	Presence	Description
`tag`	optional	If defined, the ASN.1 object is implicitly tagged with the given context specific tag.

Samples

Samples

`<null/>`
Generated ASN.1 representation:
`NULL`

`<null tag="0"/>`
Generated ASN.1 representation:
`[0] NULL`
dumpasn1 will show an error
`[0]`
`Error: Object has zero length.`

ASN.1 Constructed Types

sequence

Tag

Tag name	Enclosed content
`sequence`	Only enclosed ASN.1 tags of any type are allowed. No text or tags that insert text are permitted

Parameters

Tag parameter	Presence	Description
`includeempty`	optional	If set to `true`, the sequence will be included even if it does not contain any enclosed objects. If set to `false`, empty sequences are not included. Default behavior is to include empty sequences. This parameter is usually used when the contents of a sequence are added using conditional tags like `if` or `ifnot`.

Samples

Samples

```
<sequence>
      <objectid>1.2.840.113549.1.1.5</objectid>
      <null/>
</sequence>
```

Generated ASN.1 representation:

```
      SEQUENCE {
         OBJECT IDENTIFIER '1 2 840 113549 1 1 5'
         NULL
         }
```

set

Tag

Tag name	Enclosed content
`set`	Only enclosed ASN.1 tags of any type are allowed. No text or tags that insert text are permitted

Parameters

Tag parameter	Presence	Description
`includeempty`	optional	If set to `true`, the set will be included even if it does not contain any enclosed objects. If set to `false`, empty sets are not included. Default behavior is to include empty sets. This parameter is usually used when contents of a set are added using conditional tags like `if` or `ifnot`.

Samples

Samples

```
<set>
      <sequence>
            <ia5string tag="1"> IBM_HOST_PLATFORM </ia5string>
            <ia5string tag="2"> user1 </ia5string>
      </sequence>
</set>
```

Generated ASN.1 representation:

```
      SET {
        SEQUENCE {
          [1] 'IBM_HOST_PLATFORM'
          [2] 'user1'
          }
        }
```

explicit

Tag

Tag name	Enclosed content
`explicit`	Only enclosed ASN.1 tags of any type are allowed. No text or tags that insert text are permitted

Parameters

Tag parameter	Presence	Description
`tag`	mandatory	The explicit tag to use.
`includeempty`	optional	If set to `true`, the explicit complex type will be included even if it does not contain any enclosed objects. If set to `false`, empty explicit complex types are not included. Default behavior is to include empty explicit complex types. This parameter is usually used when contents of an explicit complex type are added using conditional tags like `if` or `ifnot`.

Samples

Samples

```
<explicit tag="0">
      <explicit tag="0">
            <ia5string tag="6">
                  http://www.keyon.ch/cacrl.crl
            </ia5string>
      </explicit>
</explicit>
```

Generated ASN.1 representation:

```
      [0] {
        [0] {
          [6]
            'http://www.keyon.ch/cacrl.crl'
          }
        }
```

ASN.1 Raw Data

This special tag allows including externally encoded ASN.1 data in a custom extension.

Tag

Tag name	Enclosed content
`raw`	Only text or tags that insert text are permitted. No enclosed ASN.1 tags are allowed. The expanded text must only contain `0..9` and `A..F` characters representing the binary data in hex format of valid ASN.1 data. Bytes may be separated with whitespace or the `:` character. If the total number of characters is odd, a `0` character is added for padding.

You must ensure that the raw data given represents a valid ASN.1 construct using DER encoding rules. If the data given is not a valid DER encoded ASN.1 data, the resulting X.509 extension will be corrupt and may cause applications that parse the extensions to crash.

Parameters

Tag parameter	Presence	Description
–	-	-

Samples

Samples

```
<raw>
      311C301A811149424D5F484F53545F504C4154464F524D82057573657231
</raw>
```

Generated ASN.1 representation:

```
      SET {
        SEQUENCE {
          [1] 'IBM_HOST_PLATFORM'
          [2] 'user1'
          }
        }
```

Unsupported ASN.1 Types

The following ASN.1 types are not supported since they are not used in X.509 certificate extensions:

- `RELATIVE-OID`
- `REAL`
- `VideotexString`
- `GraphicString`
- `ObjectDescriptor`

Using a custom extension definition to enforce subject name requirements

You can define a custom extension that is always empty and will never be included in the certificate for the sole purpose of ensuring that parts of subject name meet specific requirements. As the *if* and *ifnot* logic tags allow to specify regular expressions, one can easily implement blacklist or whitelist patterns to enforce e.g. well-formed subject names or specific domain names.

Sample

The following configuration will never add an extension as no ASN.1 tags are present and `onempty` is set to `remove`. If however the first CN element of the subject name given in the certificate request does not end in `.keyon.ch`, the certificate will not be issued (denied by policy module) and the exception text is logged in the event log and returned in the disposition message. Adding this configuration as a custom extension will thus enforce that the CN of issued certificates always have a content that ends in `.keyon.ch`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<extension oid="1.3.6.1.4.1.15486.44322931.1"
           critical="false"
           onempty="remove"
           name="CommonNameEnforcement">

    <define name="cn">
        <requestrdn oid="2.5.4.3" element="1"/>
    </define>

    <ifnot name="cn"
           match="^+*\.keyon\.ch$">
           matchoptions="IGNORECASE"
        <exception> The common name given (<insert name="cn"/>)
                    does not end in .keyon.ch </exception>
    </ifnot>

</extension>
```

You must specify an OID that meets certain requirements to be valid. You can use the OID `1.3.6.1.4.1.15486.44322931.n` (where `n`≥1) to have a valid OID if your company does not already have its own OID range which you can use.

Using a custom extension definition to enforce subjectAltName extension requirements

You can define a custom extension that is always empty and will never be included in the certificate for the sole purpose of ensuring that parts of the *subjectAltName* extension meet specific requirements. As the *if* and *ifnot* logic tags allow to specify regular expressions, one can easily implement blacklist or whitelist patterns to enforce e.g. well-formed *dNSNames* or prevent other information in the *subjectAltName* extension.

Sample

The following configuration will never add an extension as no ASN.1 tags are present and `onempty` is set to `remove`.

If however any *dNSName* entry in a requested subject alternative name extension does not end in `.keyon.ch` or an *iPAddress* is present in the extension, the certificate will not be issued (denied by policy module) and the exception text is logged in the event log and returned in the disposition message. Adding this configuration as a custom extension will thus enforce that the *subjectAltName* extension of issued certificates only have *dNSNames* entries that end in `.keyon.ch` and no *iPAddress* entries are present.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<extension oid="1.3.6.1.4.1.15486.44322931.2"
           critical="false"
           onempty="remove"
           name="SubjectAltNameEnforcement">

    <foreachsubjectaltname type="dNSName" define="myDnsName">
        <ifnot name="myDnsName" match="^.+\.keyon\.ch$"
               matchoptions="IGNORECASE">
            <exception>
                SubjectAltName policy violation: dNSName
                '<insert name="myDnsName"/>' doesn't end
                in .keyon.ch
            </exception>
        </ifnot>
    </foreachsubjectaltname>
```

```
    <foreachsubjectaltname type="ipAddress"
                           define="myIpAddress">
        <exception>
             SubjectAltName may not contain ipAddress entries
        </exception>
    </foreachsubjectaltname>

</extension>
```

Using a custom extension definition to lookup data for the subject DN

You can define a custom extension that is always empty and will never be included in the certificate for the sole purpose of looking up data in the Active Directory.

Sample

The following configuration will never add an extension as no ASN.1 tags are present and `onempty` is set to `remove`. It will however set a request attribute named `userPrincipalName` based on an Active Directory query with the `sAMAccountName` extracted from the request attribute `name` (DOMAIN\account).

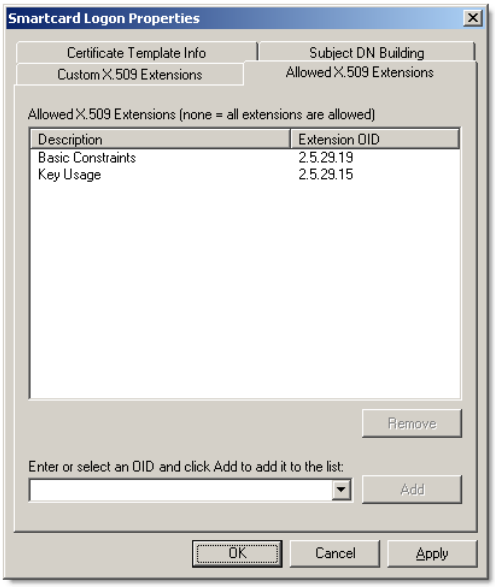
The request attribute will be only valid during the processing of the request by true-Xtender, it can however be used for the subject DN building. If the Active Directory lookup fails, an exception will be thrown.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<extension oid="1.3.6.1.4.1.15486.44322931.3"
           critical="false"
           onempty="remove"
           name="AD Lookup of userPrincipalName">

    <define name="sAMAccountName" match="^.+\\(.+)$">
        <requestattribute name="name" ignorecase="true"/>
    </define>

    <ifnot name="sAMAccountName">
        <exception>
            The sAMAccountName could not be determined from
            the request
        </exception>
    </ifnot>
```

```
    <define name="upn">
        <adsquery
           attribute="userPrincipalName"
           searchfilter="(&amp;
                   (sAMAccountName={sAMAccountName})
                   (objectCategory=person)
                   (objectClass=user))"/>
    </define>

    <ifnot name="upn">
        <exception>
            The UPN could not be found in the Active Directory
        </exception>
    </ifnot>

    <definerequestattribute name="userPrincipalName">
        <insert name="upn"/>
    </definerequestattribute>

</extension>
```

Allowed X.509 Extensions

The *Allowed X.509 Extensions* tab allows restricting the X.509 certificate extensions that will be included in the certificate. Extensions that are not in the list will not be included in the certificate. This functionality can be used to suppress Microsoft specific extensions like *Certificate Template Name*.

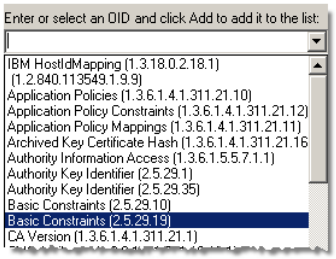


If you do not specify any extensions, all X.509 extensions added by the Microsoft policy module will be present in the certificate. If you define custom extensions, the custom extensions are automatically added to the allowed extensions list unless the list is empty.

Some Microsoft specific extensions are required for automated processes like renewal when the Enterprise functionality of Certificate Services is used. If those extensions are suppressed, such processes may no longer work.

Adding well known extensions

To add a well-known extension, click on the drop list button under *Enter or select an OID and click Add to add it to the list* and select the desired extension:


Enter or select an OID and click Add to add it to the list:
IBM HostIdMapping (1.3.18.0.2.18.1)
(1.2.840.113549.1.9.9)
Application Policies (1.3.6.1.4.1.311.21.10)
Application Policy Constraints (1.3.6.1.4.1.311.21.12)
Application Policy Mappings (1.3.6.1.4.1.311.21.11)
Archived Key Certificate Hash (1.3.6.1.4.1.311.21.16
Authority Information Access (1.3.6.1.5.5.7.1.1)
Authority Key Identifier (2.5.29.1)
Authority Key Identifier (2.5.29.35)
Basic Constraints (2.5.29.10)
Basic Constraints (2.5.29.19)
CA Version (1.3.6.1.4.1.311.21.1)

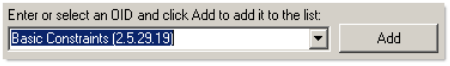
The list shows all OIDs known to Microsoft Windows as well as any custom extensions defined for this certificate template with the keyon / true-Xtender.

Note that some extensions are present with multiple OIDs (e.g. Basic Constraints) due to historical reasons. Make sure you use the correct OID if multiple OIDs are available for a given extension.

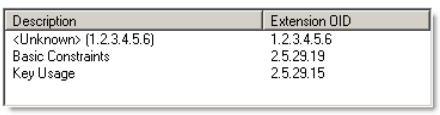
The following list shows the current OID for some common extensions with multiple OIDs available:

Extension	Current OID
Authority Key Identifier	2.5.29.35
Basic Constraints	2.5.29.10
Issuer Alternative Name	2.5.29.18
Policy Mappings	2.5.29.33
Subject Alternative Name	2.5.29.17

The selected extension is automatically inserted into the edit field after the selection:

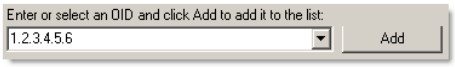


After clicking Add, the extension is added to the list:

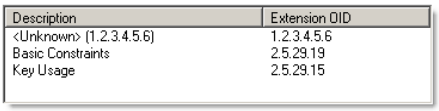


Adding extensions by OID

Simply enter the OID of the extension under *Enter or select an OID and click Add to add it to the list*:



After clicking Add, the extension with the entered OID is added to the list. If the OID is not known to Microsoft Windows, the description of this OID will be <Unknown> (*OID*):

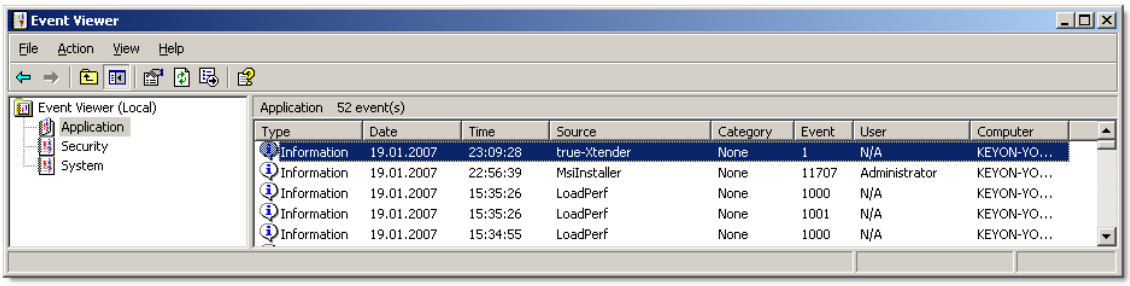


Removing extensions from the list

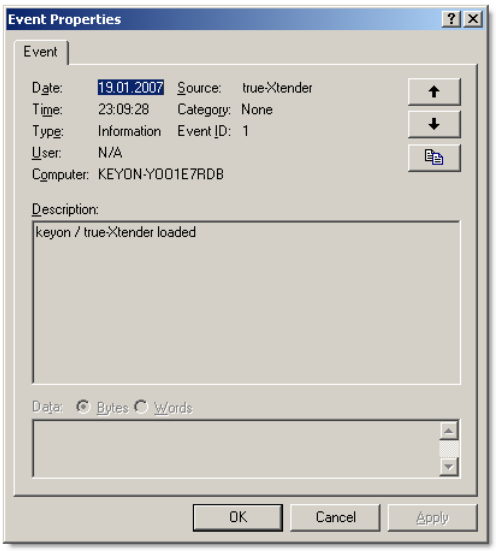
Simply select the extension in the list and click the *Remove* button to remove an extension from the list.

Event Log

The keyon / true-Xtender logs all events in the Windows Application Event Log:


Event Viewer
File Action View Help
Event Viewer (Local)
Application
Security
System
Application 52 event(s)
Type Date Time Source Category Event User Computer
Information 19.01.2007 23:09:28 true-Xtender None 1 N/A KEYON-YO...
Information 19.01.2007 22:56:39 MsiInstaller None 11707 Administrator KEYON-YO...
Information 19.01.2007 15:35:26 LoadPerf None 1000 N/A KEYON-YO...
Information 19.01.2007 15:35:26 LoadPerf None 1001 N/A KEYON-YO...
Information 19.01.2007 15:34:55 LoadPerf None 1000 N/A KEYON-YO...

All events logged by keyon / true-Xtender use *true-Xtender* as the source and can thus be easily filtered out in the vent viewer:



Events IDs used

The following table lists the events that can be logged:

ID	Type	Description
1	Information	keyon / true-Xtender loaded
2	Information	keyon / true-Xtender unloaded
3	Error	Unable to load certificate templates from Active Directory. Please restart certificate services.
4	Error	Unable to load Microsoft Certificate Policy COM object
5	Information	Changed DN from '%1' to '%2' based on configuration for template '%3'
6	Error	Microsoft policy module was unable to pre-process the request.%n Windows error: %1%n COM error info: %2%n
7	Error	Unable to get CLSID_CCertServerPolicy: %1
8	Error	Template error: %1
9	Error	Windows error: %1
10	Error	Failed to change DN %1
11	Error	Template %1 not found in Active Directory. Unable to process request.
12	Error	Unable to process request. Microsoft Certificate Policy COM object not found.
13	Error	Unable to add custom X.509 extension: %1
14	Error	Unable to build new DN: %1

Regular expressions

The regular expression engine used by true-Xtender supports a Perl compatible pattern syntax. The basic patterns are thus the same as used by most regular expression engines.

For an explanation of regular expressions, see [REGEXP-1], a tutorial can be found at [REGEXP-2]. Visit [REGEXP-3] for an online library of common regular expressions.

Syntax

Common characters

Alphanumeric characters, the underscore and punctuations with no special definition (see special characters) match themselves. If all the characters in a pattern are common characters, the match operation is just a check if the pattern string occurs in the given text.

Note

- Character matching is case sensitive unless `IGNORECASE` mode is set.

Special characters

Non printable characters

Nonprintable characters (e.g. newline, tab etc.) are specified as follows:

Escape	Description
`\a`	Bell = `\x07`
`\f`	Page = `\x0C`
`\n`	New line = `\x0A`
`\r`	Return = `\x0D`
`\t`	Tab = `\x09`
`\v`	Vertical tab = `\x0B`
`\e`	ESC = `\x1B`
`\x20`	Use two digit hex number, match the char
`\u002B`	Use four digit hex number, match the char
`\x{20A060}`	Use any hex number, match the char

Characters for pattern expressions

The following characters are used for building the pattern:

Character	Description
^	Matches the beginning of the string. Use \^ to match ^ itself
$	Matches the end of the string. Use \$ to match $ itself
( )	Grouping. Use \(and \) to match (and)
[ ]	Character class. Use \[and \] to match [and]
{ }	Define quantifiers. Use \{ and \} to match { and }
.	Match any character except newline (\n). Use \. to match . itself
?	Let sub pattern match 0 or 1 time. Use \? to match ? itself
+	Let sub pattern match at least 1 times. Use \+ to match + itself
*	Let sub pattern match any times. Use * to match * itself
|	Alternation. Use | to match | itself

Character sets

Character set can match any of the characters in the set.

Standard character sets

Set	Description
.	The dot can match any single character except the newline (\n)
\w	Match any alphanumeric character and the underscore
\W	Match any character not in \w
\s	Match space, new-line, return etc. (any space character)
\S	Match any character not in \s
\d	Match any number, 0~9
\D	Match any character not in \d
[:alnum:]	Match any alphanumeric character (A-Z, a-z, 0-9)
[:alpha:]	Match any alpha character (A-Z, a-z)
[:ascii:]	Match any ASCII character (\x00-\x7F)
[:cntrl:]	Match any control character (\x00-\x1F, \x7F)

Set	Description
[:digit:]	Match any number (0-9)
[:print:]	Match any printable ASCII character (\x20-\x7E)
[:space:]	Match any space character (\x09-\x0D, \x20)
[:graph:]	Match any printable ASCII character, excluding space (\x21-\x7E)
[:lower:]	Match any lowercase alpha character (a-z)
[:punct:]	Match any printable ASCII character excluding [:alnum:]
[:upper:]	Match any uppercase alpha character (A-Z)
[:xdigit:]	Match any hex number (0-9, A-F, a-f)
[:blank:]	Match space or tab (\x20, \x09)

Notes

- If [: is followed by ^, it will invert the match. All characters but the one in the set will match in this case.
- If the match mode is SINGLELINE, the dot can match any character including new-line.

Custom defined character sets

Custom character sets will match any of the characters defined using [...]. Use [^...] to define a negative set which will match any character not in the set.

Notes

- Special characters will lose their special meaning when part of a character set.
- Standard character sets can be added into custom character sets. For example: [\d.\-+] will match any of 0-9, ., -, and +. POSIX character set can be added into custom character sets set as well.
- Use the minus character - to define a range of characters to add to the set. For example: [\dA-Fa-f], will match any of 0123456789, ABCDEF and abcdef.
- Character sets are case sensitive even if IGNORECASE mode is used.

Repeat quantifiers

Match another expression multiple times. By default, a quantified sub pattern is greedy, i.e. it will match as many times as possible (given a particular starting location) while still allowing the rest of the pattern to match.

Standard quantifiers

Quantifier	Description
`?`	Match `0` or `1` time. Equivalent to `{0,1}`
`+`	Match `1` or more times. Equivalent to `{1,}`
`*`	Match `0` or more times. Equivalent to `{0,}`
`{n}`	Match `n` times. Example: `\w{2}` is equivalent to `\w\w`
`{m,n}`	Match least `m` times and at most `n` times. Example: `ba{1,3}` will match `ba`, `baa` and `baaa`.
`{m,}`	Match least `m` times. Example: `\w\d{2,}` will match `a12`, `x456` ...

Reluctant quantifiers

If a quantifier is followed by a question mark (`?`), it becomes a reluctant quantifier. Reluctant quantifiers will match the minimum number of times possible.

Quantifier	Description
`{m,n}?`	Match only `m` times if possible, at most `n` times.
`{m,}?`	Match only `m` times if possible, can match as many times as necessary.
`??`	Match `0` time if possible, at most `1` time. Equivalent to `{0, 1}?`
`+?`	Match only `1` time if possible, can match as many times as necessary, `{1, }?`
`*?`	Match `0` time if possible, can match as many times as necessary, `{0, }?`

Possessive quantifiers

If a quantifier is followed by a plus (+), it becomes a possessive quantifier. Possessive quantifiers will greedily match as much as they can and do not back off, even when doing so would allow the overall match to succeed.

Quantifier	Description
`{m,n}+`	Match `n` times if possible, `m` times at least.
`{m,}+`	Match as many times as possible, `m` times at least.
`?+`	Match `1` time if possible, match `0` time if could not, equivalent to `{0,1}+`
`++`	Match as many as possible, `1` times least. Equivalent to `{1, }+`
`*+`	Match as many as possible, `0` times if could not match. Equivalent to `{0, }+`

Character boundaries

The following elements are used to specify a condition for a position in the string.

Condition	Description
`^`	Current position must be the beginning of text
`$`	Current position must be the end of text
`\b`	Current position must be a word boundary

Notes

- If the match mode is `MULTILINE`, `^` will match line begin and `$` can match line end.
- If `\b` is added to a character set it will stand for the backspace character (ASCII = 8).

Alternative expressions

Use "`|`" to match any one of multiple alternative expressions. The engine will try alternative expressions from left to right.

Grouping

Use brackets "`(...)`" to enclose sub expressions into a single element. Sub expressions in it will be a whole part when it is quantified. The first captured group can be retrieved using the define tag.

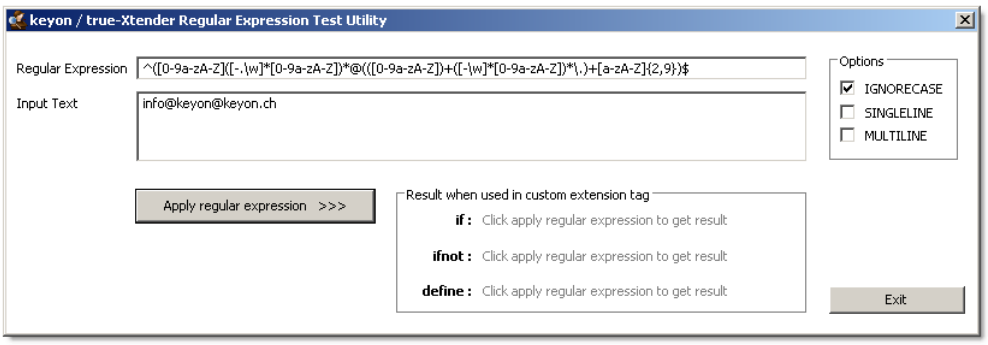
Options

The following options can be specified to change the behavior of the

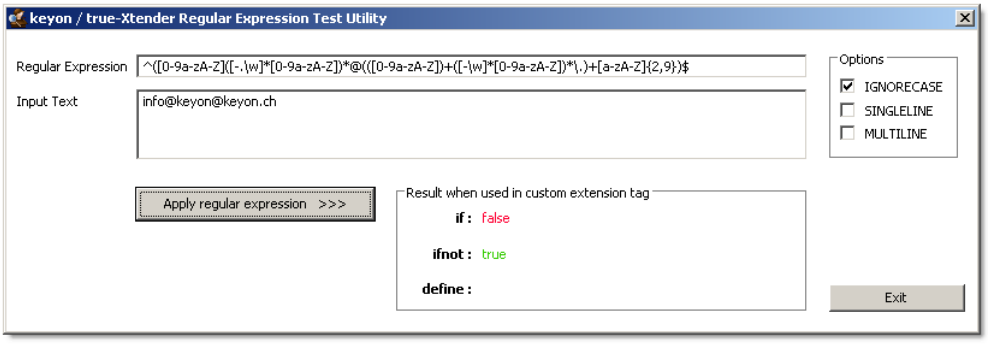
Option	Description
`IGNORECASE`	By default, regular expression is case sensitive. Apply this mode to match case insensitively. But character sets are always case sensitive.
`SINGLELINE`	By default, the dot can match any character except newline (`\n`). `SINGLELINE` mode will let the dot to match any character.
`MULTILINE`	Change `^` and `$` from matching the start or end of the string to matching the start or end of any line anywhere within the string. `SINGLELINE` and `MULTILINE` only sound mutually exclusive. They have different function at different places, so they can be used together. The assertions `\A` and `\Z` are used to match only the beginning and end of the string respectively, regardless of whether the `MULTILINE` flag has been specified.

Regular Expression Test Utility

The true-Xtender ships with a utility for the design and test of regular expressions. After starting the utility, the regular expression, the desired options and the text to match are entered into the appropriate fields:

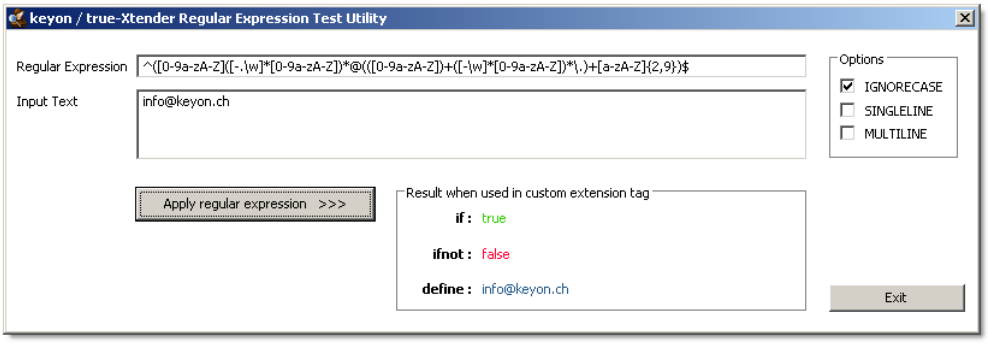


By clicking *Apply regular expression* >>>, the input text is matched against the regular expression and the result when used in the different tags (*if*, *ifnot*, *define*) is shown:



In this case false for the *if* tag means that the enclosed content would not be considered when processing the extension template as the match failed. As nothing was matched, *define* would assign an empty string to the variable. (The sample shows a regular expression which checks for a valid email address.)

You can change the regular expression or the input text and click *Apply regular expression* >>> to execute the match with the new data:



Common Problems

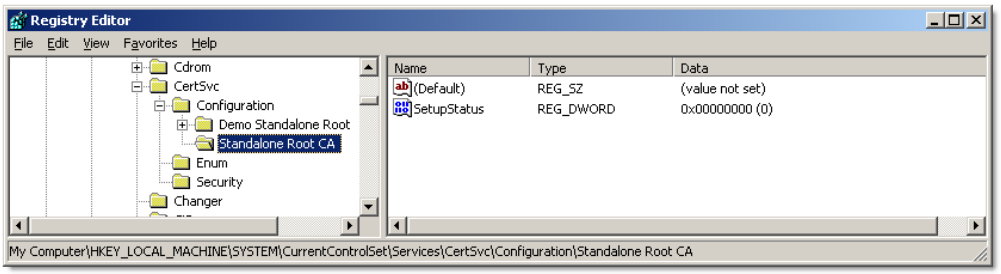
The following list shows some common errors when using keyon / true-Xtender.

Unable to configure true-Xtender using the configuration application after installation

When the true-Xtender configuration is started using the supplied configuration application *Configure true-Xtender*, you may encounter the following error:

This problem occurs if Certificate Services was only partially installed, e.g. because the creation of the CA was aborted. If Certificate Services is removed and added again but using a different CA name, the reminders of the first CA causes the problem.

The registry shows two CA entries (or even more) in this case:



Resolution

Use the registry editor `regedit.exe` to delete the registry key of the previous CA under

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\
```

Reference

Tools

The following tools can be used to dump and examine X.509 certificates to verify the correct implementation of the certificate customizations:

dumpasn1 — ASN.1 dump utility
http://www.cs.auckland.ac.nz/~pgut001/

GUIdumpASN — GUI version of the ASN.1 dump utility
http://www.geminisecurity.com/guidumpasn.html

openssl — Standard to dump and convert X.509 certificates
http://www.openssl.org

Literature

The following documents are useful when specifying X.509 certificates and extensions:

[RFC3280] Internet X.509 Public Key Infrastructure / Certificate and Certificate Revocation List (CRL) Profile
http://www.ietf.org/rfc/rfc3280.txt

[ASN-1] ITU-T X.690 SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS
OSI networking and system aspects – Abstract Syntax Notation One (ASN.1)
Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

[ASN-2] "ASN.1 - Communication between heterogeneous systems" by Olivier Dubuisson
http://asn1.elibel.tm.fr/en/book/
http://www.oss.com/asn1/booksintro.html

[ASN-3] ASN.1 Complete by Prof John Larmouth
http://www.oss.com/asn1/larmouth.html

[REGEXP-1] Explanation of regular expressions
http://en.wikipedia.org/wiki/Regular_expression

[REGEXP-2] Tutorial for regular expressions as supported by true-Xtender
http://www.regexlab.com/en/regref.htm

[REGEXP-3] Library of regular expressions
http://regexlib.com/

[ADS-1] LDAP ADsPath specification
http://msdn.microsoft.com/en-us/library/aa746384%28v=VS.85%29.aspx

[ADS-2] LDAP Search Filter Syntax
http://msdn.microsoft.com/en-us/library/aa746475%28v=VS.85%29.aspx

[RFC2254] The String Representation of LDAP Search Filters
http://www.ietf.org/rfc/rfc2254.txt

[MS-WCCE] Windows Client Certificate Enrollment Protocol Specification
http://msdn.microsoft.com/en-us/library/cc249879%28v=prot.13%29.aspx

Links

OID repositories http://www.alvestrand.no/objectid/top.html
http://asn1.elibel.tm.fr/oid/index.htm